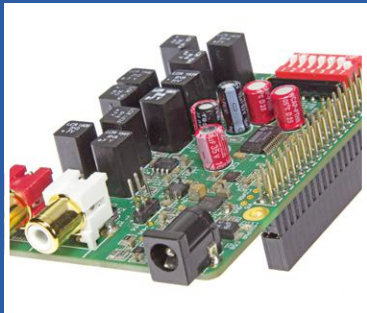




elektor

LEARN → DESIGN → SHARE



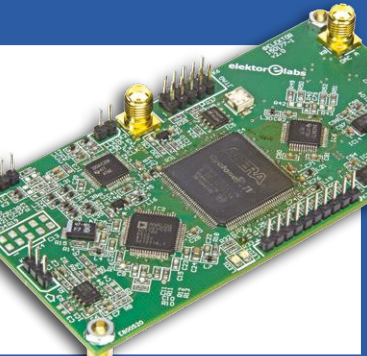
Audio DAC for Raspberry Pi

A network audio player with Volumio



PCBs — Printed, not Etched

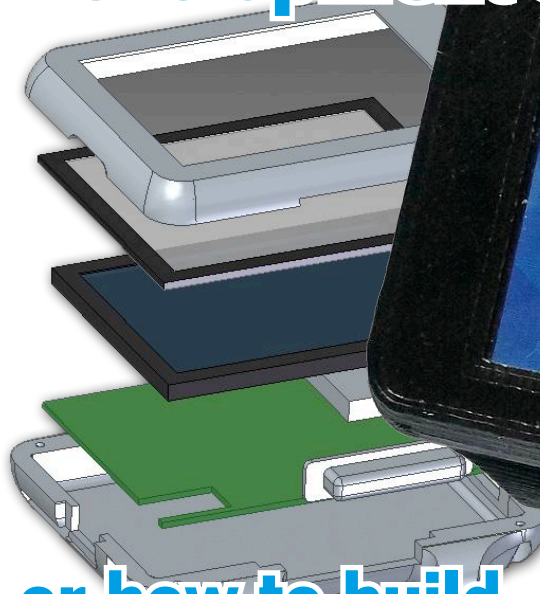
the Voltera V-One PCB printer and reflow soldering station



FPGA-DSP Board for Narrowband SDR

A fully programmable receiver/transmitter baseboard

nWatch a Wearable Development System

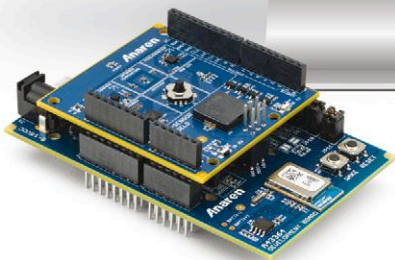


or how to build your own smart watch

2-Terminal Dimmer • Audio DAC for Raspberry Pi • **BBC micro:bit for Electronicists (2)** • **Components in Space** • Data acquisition and oscilloscope functions • **Electronics in Orbit** • **Elektor Labs Pipeline** • **Err-lectronics** • **FPGA-DSP Board for Narrowband SDR** • **Get 5 V from One Exhausted Alkaline Cell** • **Homelab Helicopter** • **Keep Cool with an ATtiny** • **Making Lights Meaningful** • **May I have the menu, please?** • **Mercury Coulometers** • **MicroPython and the pyboard** • **nWatch, a Wearable Development System** • **PCB Design is Not Easy** • **PCBs — Printed, not Etched** • **RF Step Attenuator** • **Sea Murmur Simulator** • **Sensors Make Sense (4)** • **Smart Devices** • **Sniffer using RFM12 Radio Module** • **Test Pattern Generator using a Digital Camera** • **Tektronix 556 and 565 Dual-beam Oscilloscopes** • **The ESP8266's Big Brother** • **The I²C Bus** • **Ukele-LED** • **Zenerdoku (an extraordinary Hexadoku)** • **And more**

Take Off and Soar

With the Anaren Atmosphere 2.0 IoT Development environment and Atmosphere Cloud™ hosting, – plus – the new A43364 Wi-Fi development board, pilot your IoT project from concept to cloud connectivity at Mach speed. No delays. No runaway costs. First class all the way.



www.anaren.com/air/win

* Must be 18 to participate. No purchase necessary. North America only. Void where prohibited. Other restrictions apply.



Scan the code to learn more and register for a chance to win a free Atmosphere MSDK!*

Only the Atmosphere 2.0 IoT Development environment with Atmosphere Cloud™ hosting gives you the fastest trajectory for developing connected, embedded Wi-Fi applications, complete with mobile apps and cloud support. Think. Build. Connect.

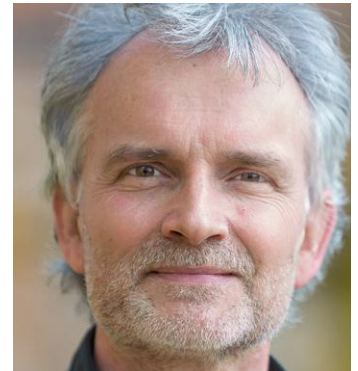


Elektor Magazine

Edition 4/2017

Volume 43, No. 484

July & August 2017



ISSN 1757-0875 (UK / US / ROW distribution)

www.elektor.com

www.elektormagazine.com

Elektor Magazine, English edition
is published 6 times a year by

Elektor International Media

78 York Street

London W1H 1DP

United Kingdom

Phone: (+44) (0)20 7692 8344

Head Office:

Elektor International Media b.v.

PO Box 11

NL-6114-ZG Susteren

The Netherlands

Phone: (+31) 46 4389444

Fax: (+31) 46 4370161

Memberships:

Please use London address

E-mail: service@elektor.com

www.elektor.com/memberships

Advertising & Sponsoring:

Johan Dijk

Phone: +31 6 15894245

E-mail: johan.dijk@eimworld.com

www.elektor.com/advertising

Advertising rates and terms available on
request.

Copyright Notice

The circuits described in this magazine are for domestic and educational use only. All drawings, photographs, printed circuit board layouts, programmed integrated circuits, disks, CD-ROMs, DVDs, software carriers, and article texts published in our books and magazines (other than third-party advertisements) are copyright Elektor International Media b.v. and may not be reproduced or transmitted in any form or by any means, including photocopying, scanning and recording, in whole or in part without prior written permission from the Publisher. Such written permission must also be obtained before any part of this publication is stored in a retrieval system of any nature. Patent protection may exist in respect of circuits, devices, components etc. described in this magazine. The Publisher does not accept responsibility for failing to identify such patent(s) or other protection. The Publisher disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from schematics, descriptions or information published in or in relation with Elektor magazine.

© Elektor International Media b.v. 2017

www.eimworld.com

Printed in the Netherlands

You can leave your hat on

Today, if you see a board with a pinheader along the sides, or its female counterpart the pinheader socket, in single- or double-row guise, most likely you are looking at an extension board for one of the popular DIY embedded platforms like Raspberry Pi, Arduino, BeagleBone and many others. These stack-on boards go by trendy names like hat (RPI), shield (Arduino), or cape (BB). The name 'hat' is a good find as besides being an acronym for Hardware Added on Top; it doubles nicely as the device you put on top of your RPi board. The 'shield' I find a little puzzling as I was taught at school this is a protective device designed to keep the bearer from being hurt by sharp or heavy objects hurled or poked at him in battle situations. Defensive in essence, not aggressive. One wonders what Arduino has to ward off as it is doing the opposite, I mean attracting, like a magnet, all sorts of external hardware, I/O, and sensor activity in general? The cape, too, is a weird designation as I am challenged to associate any dog let alone its bone with a cape, be it referring to a sleeveless cloak or a promontory. And neither is that friendly looking doggie in beaglebone.org's logo ever seen wearing the garment, or high on a windswept rock peering over the 'vast waters — of the petrel and the porpoise'. Puzzling.

Shields, capes or hats come in a bewildering variety covering just about anything you'd want to do learn from, or inflict on, the real world (your I and O respectively). Check out the .org websites associated with each of The Three Platforms, and the .com sites of a flood of board retailers and you'll find that most, if not all, plug-on boards cost more than the computerette they sit on. Users seem to be happy with that simply because the hat/cape/shield is a fit-and-forget device that allows people to concentrate on what the system, rather than the MCU alone, is actually doing. Hats off to the meticulously designed RPi Audio DAC on page 22.

Jan Buiting, Editor-in-Chief

The Circuit

Editor-in-Chief:

Jan Buiting

Deputy Editors:

Thijs Beckers, Clemens Valens

Translators:

David Ashton, Jan Buiting, Martin Cooke,
Ken Cox, Arthur deBeun, Andrew Emmerson,
Tony Marsden, Mark Owen, Julian Rivers

Membership Manager:

Raoul Morreau

International Editorial Staff:

Thijs Beckers, Mariline Thiebaut-Brodier
Denis Meyer, Jens Nickel

Laboratory Staff:

Ton Giesberts, Luc Lemmens,
Clemens Valens, Jan Visser

Graphic Design & Prepress:

Giel Dols

Publisher:

Don Akkermans



This Edition

Volume 43 – Edition 4/2017

No. 484 July & August 2017



Regulars

- 13 Elektor Connexions**
- 59 Homelab Helicopter**
- 68 Err-electronics**
Corrections, Updates and Feedback to published articles
- 79 Elektor Labs Pipeline**
- 85 Q & A**
Components in Space
- 100 Retronics: Tektronix 556 and 565 Dual-beam Oscilloscopes**
The Triple-L beasts: large, loud, lumpy
- 107 Tips & Tricks**
Test Pattern Generator using a Digital Camera
- 117 Peculiar Parts, the series**
Mercury Coulometers
- 126 Elektor Ethics**
Smart Devices
- 128 Elektor Store**
- 130 Zenerdoku (an extraordinary Hexadoku)**



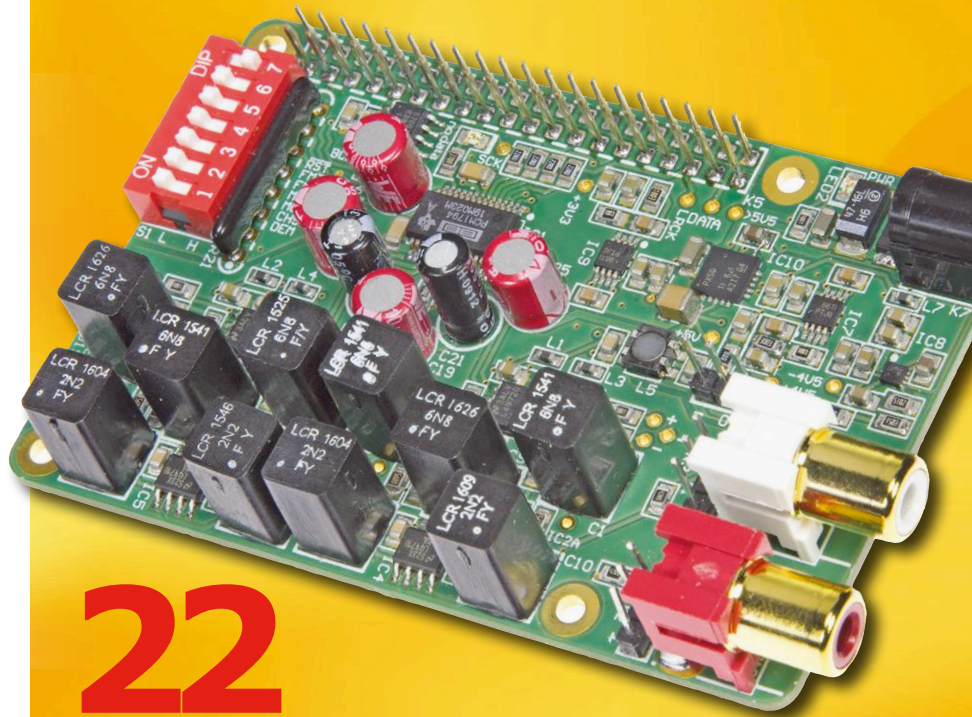
Features

- 6 PCB Design is Not Easy**
The art of laying traces
- 44 PCBs – Printed, not Etched**
Presenting the Voltera V-One PCB printer and reflow soldering station
- 94 Electronics in Orbit**
Space science for all of us
- 99 “May I have the menu, please?”**



Projects

- 14 2-Terminal Dimmer**
controls LEDs, lamps and heaters
- 18 RF Step Attenuator**
Adjustable attenuation gives precise signal levels
- 22 Audio DAC for Raspberry Pi**
A network audio player with Volumio
- 30 BBC micro:bit for Electronicists (2)**
Data acquisition and oscilloscope functions
- 35 Sea Murmur Simulator**
Gentle noise to help you go to sleep
- 38 The I²C Bus**
Part 2: using the bus with a microcontroller



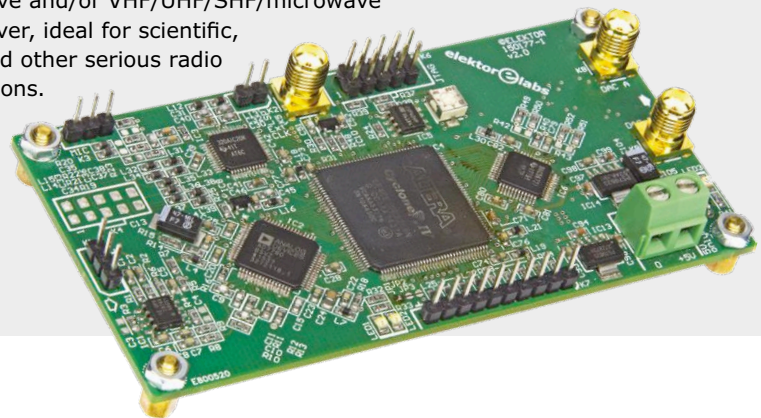
22

FPGA-DSP Board for Narrowband SDR

A fully programmable receiver/transmitter baseboard

Presented here is an FPGA-based digital signal processing board featuring everything necessary to do baseband processing of traditional narrowband modes like SSB/CW and AM. It solves many of the issues and limitations seen in simpler SDR platforms. Capable of transmitting as well as receiving, it provides a perfect foundation for a shortwave and/or VHF/UHF/SHF/microwave transceiver, ideal for scientific, ham, and other serious radio applications.

70



44



Audio DAC for Raspberry Pi

A network audio player with Volumio

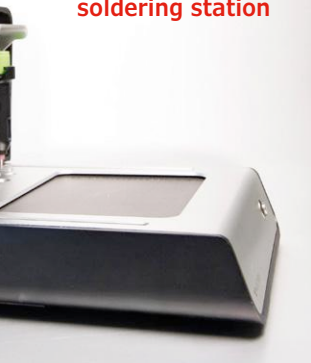
The ESP8266's Big Brother

Getting started with ESP32 and the
Arduino IDE



The ESP8266 from Espressif has made waves in the maker community with its low price: you can get one from AliExpress for just two or three euros/dollars/pounds. Its big brother, the ESP32, is the result of Espressif's efforts to make their WLAN SoCs fit for the future. Along with a dual-core processor, it has onboard Bluetooth, more RAM, and IO extensions. In this article we talk about programming the ESP32 with the Arduino IDE. In the next issue of Elektor we will turn our attention to the native development tool IDF.

Presenting the
Voltera V-One
PCB printer
and reflow
soldering station



- 48 nWatch, a Wearable Development System**
or how to build your own smart watch
- 62 Get 5 V from One Exhausted Alkaline Cell**
Juice it to the Max!
- 70 FPGA-DSP Board for Narrowband SDR**
A fully programmable receiver/transmitter baseboard
- 80 MicroPython and the pyboard**
From blinking LED to ... webserver blinking an LED

- 88 The ESP8266's Big Brother**
Getting started with ESP32 and the Arduino IDE
- 104 Sniffer using RFM12 Radio Module**
spy on the ether
- 108 Sensors Make Sense (4)**
For Arduino and more
- 114 Keep Cool with an ATtiny**
Temperature-dependent fan control
- 118 Ukele-LED**
Serially addressable LEDs as ukulele learning tool
- 122 Making Lights Meaningful**
For all model aircraft

Next Editions

Elektor Magazine 5 / 2017

Tangible Prototypes for Tabula • LoRa with ST Boards • CPLD Board as DIL Component • FFT Scopes in Practice • ESP32 with IDF • Datalogger for Heating Systems • RPi Zero W Review • Function Generators • Flexible IoT sensor interface with GoNotify • Scoreboard with 7-segment Displays • 12-V LED Driver • Ultrasonic Sound Generator • Universal I2C Bus Isolator and Level Adapter • Temi Temperature & Humidity Sensor • Spy Camera Detector • and much more.

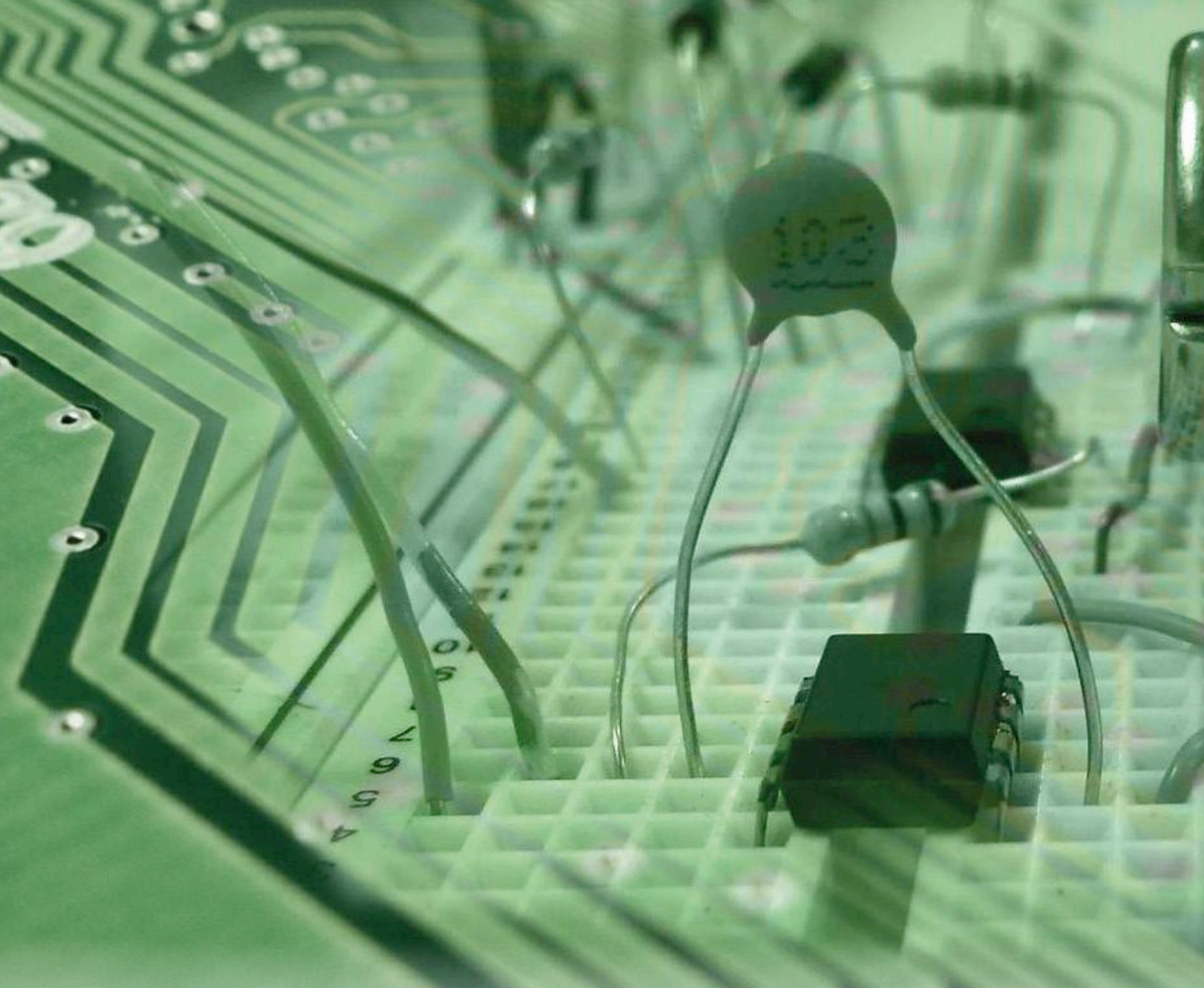
Elektor Magazine edition 5 / 2017 covering September & October is published on 15 August 2017. Delivery of printed copies to Elektor Gold Members is subject to transport. Contents and article titles subject to change.

Elektor Business Magazine 4 / 2017

Focus on: LEDs, Displays & Drivers

AS7221 Spectral Tuning IoT Smart Lighting Manager (ams) • Smart City Transformation (Schröder) • GP-LV-32D Driver Series (GlacialPower) • Lighting Audit, Spectrum Mixer (GL Spectrosoft) • MiP-Displays in Medical Applications (Sharp Devices Europe) • High-power Colour and White LED Emitters (LED Engin) • Lights for the 'Oasis' Rinspeed concept car (OSRAM) • and more.

Elektor Business Magazine edition 4/2017 is published on 15 July 2017. Contents and article titles subject to change.



PCB Design is Not Easy

The art of laying traces

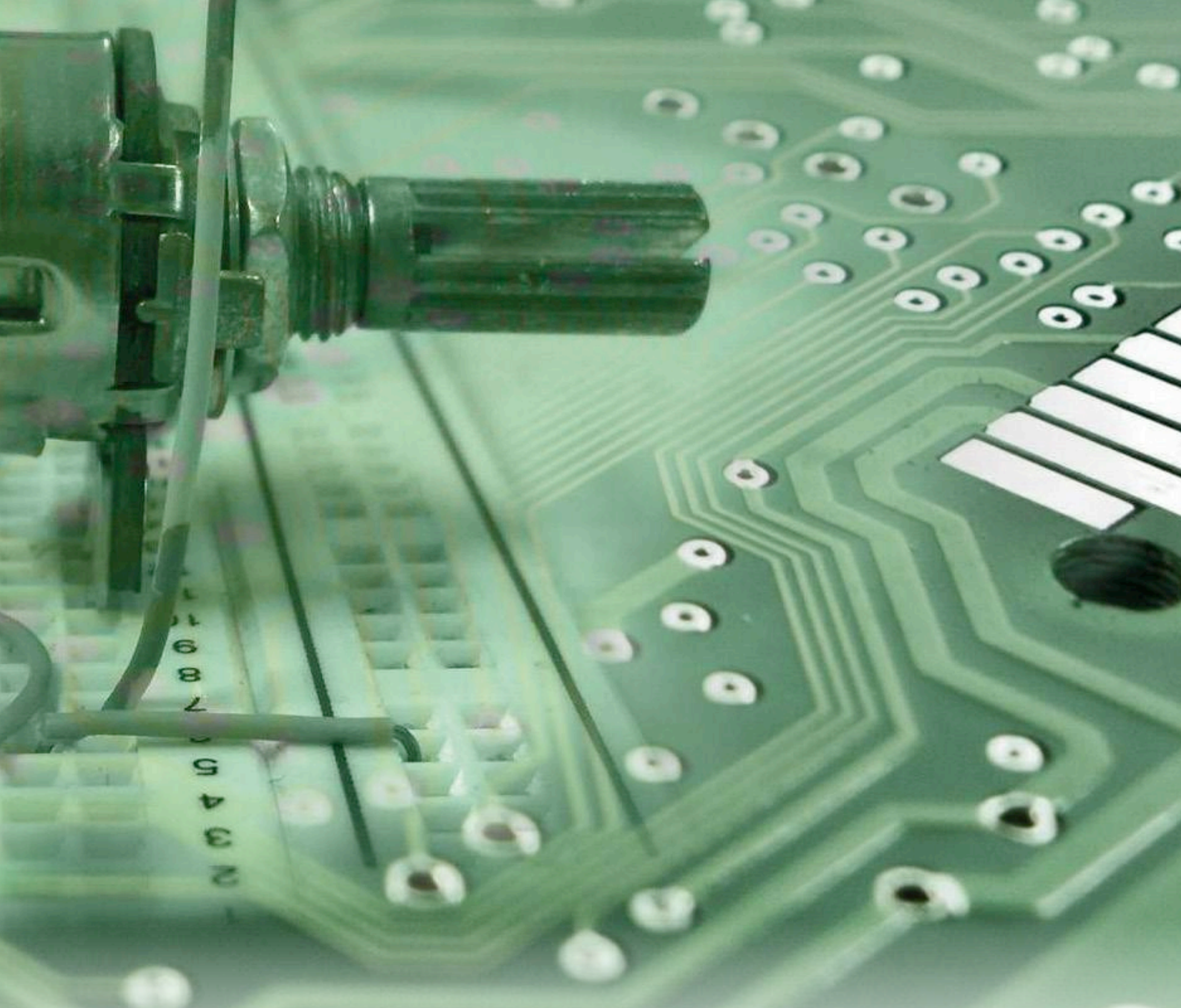
By **Clemens Valens** (Elektor Labs)

Even if such a naive approach may work for simple, low-power and low-frequency circuits, it is wrong. Simple circuits need good PCBs too. Good PCBs not only make a circuit work reliably and as intended, they also limit EMC problems as much as possible, they provide test points for system assembly and repair, and they are easy to fit in the final application. Because that is what a circuit board really is, a component in a system, and as such it has to be as good as any other component used in that system. A PCB design isn't intended to please its designer; it must please the end user, whoever that may be (**Figure 1**). In what follows, acronyms and jargon are used. Refer to the

Glossary further on in this article to better understand the text. Also, even though this article is quite long, it is far from complete. PCB design is simply too vast a subject to be treated comprehensively in one article.

System integration

The first step — skipped by most people — involves studying how the board is supposed to be integrated into the system. What shape should the board have? Where should the mounting holes be? How many? And the cables and wires to and from the board? Where do they come from, where do they go? Are there any height limitations? Possible heating issues? How is the user going to interact with the board? Will there be plug-gable cables? On the rear or on the front, on the side maybe?



To many people, designing a printed circuit board (PCB) is a mere detail or afterthought. The circuit has been tested thoroughly and then, without thinking about it too much, they plunk it onto a board to make it easier to move around without wires coming loose or components dropping off. How wrong they are...

Rotary encoder or pushbuttons? Display? LEDs? Are there possible issues with the material the enclosure is made of? Oh yes — another afterthought — what about the power supply? Even if the system consists merely of your circuit mounted in an enclosure, then, before you set out designing a PCB for the circuit, choose a suitable enclosure and adapt the PCB's size and shape to it.

Unless you have excellent metal, wood and plastic working skills, or have access to CNC and laser cutting equipment, try to limit the mechanical work to a minimum.

Board manufacturing

Another important question is: how is the board going to be made? Are you planning on etching it yourself at home? Are you

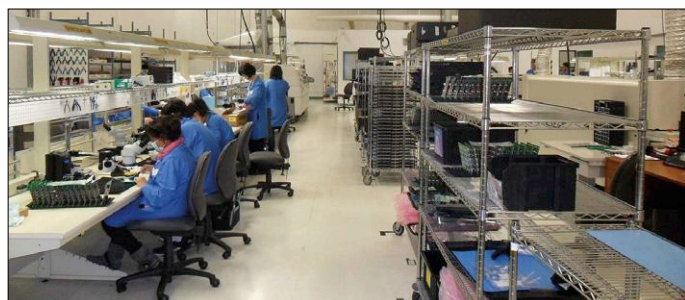


Figure 1. Your board may be built in a factory like this, somewhere far away in your perception. Therefore, for the best chances of success, design it in such a way that it can be done with as little explaining as possible.

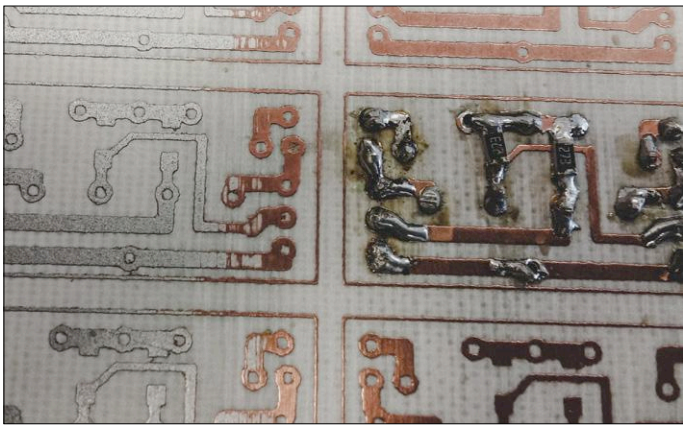
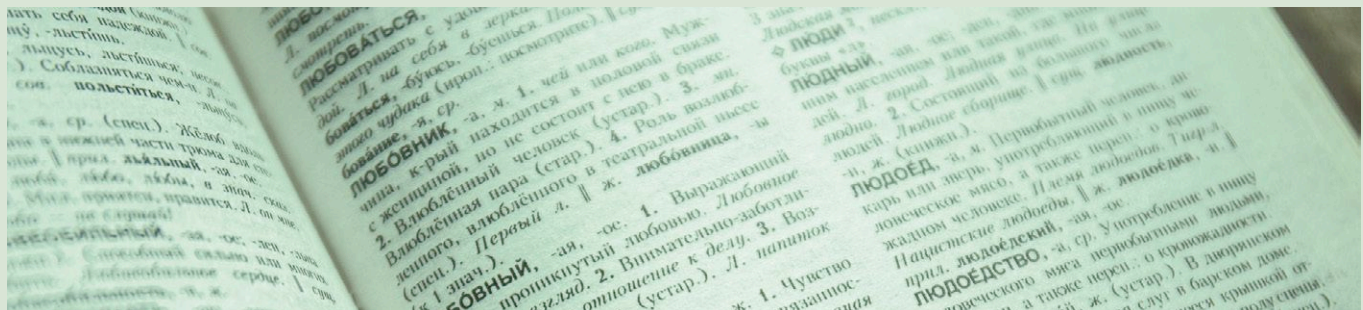


Figure 2. A home etched panel with many holes left to drill, and panel cutting still to be done. Is it really worth it to do this yourself?

any good at it? If not, you may want to avoid vias, and stick to one layer. Double-sided boards require excellent alignment of the top and bottom films, and metalizing vias can be a lot of work, especially when there are a lot. Having said that, many home etchers achieve great results with double-sided boards; it is mostly a matter of skill and experience.

You may want to stay away from ground and other copper pours (planes) since you don't have a solder mask to avoid solder bridges. Finding and cutting almost invisible shorts is an unpleasant job as well. Maybe use thicker traces to prevent them from getting eaten away by the etching fluid? To avoid copper bridges, don't place traces too close to each other. Use larger pads because you're not so good at drilling (**Figure 2**)? A PCB engraving machine has similar requirements but, unless you are a very good home etcher, it has higher precision. The trace density can be higher, but not too high either. If the mill-

A Small Glossary of PCB Design



Auto-router: Holy Grail of PCB design software developers

Bottom: the lowest layer of the board stack, also still commonly known as solder side

Class: resolution or density of a PCB; the higher the class, the smaller PCB elements and clearances are allowed to be

Clearance: distance between two or more PCB elements

DRC: design rule check, to verify that PCB elements respect a set of required design parameters like minimal trace width, minimal drill size, minimum clearance between pads, etc.

ERC: electrical rule check, to verify that nets are connected, not overlapping, no shorts exist due to copper pours, etc.

Excellon: data format for CNC drill and route machines

Fiducial: special mark on a film, mask, board, panel, etc., to help aligning them with cameras, stencils, machines, each other, etc.

Gerber: ASCII vector data format for two-dimensional, two-color images

Heat relief: pad-copper (plane or trace) connection preventing solder heat from being absorbed by the copper

IAR: inner annular ring

Layer: surface carrying PCB elements like traces, pours and components

Metalized: see Plated

Mil: one thousandth of an inch

Millimeter: one thousandth of a meter

Net: a desired connection between two or more pins

OAR: outer annular ring

Pad: PCB element for connecting a pin

PCB: printed circuit board

PCB element: object printed on a layer, includes the board outline

Pin: connection point of a component like a pad, pin, lead, etc.

Plane: large copper area; called power plane when connected to ground or a supply voltage

Plated: covered with a conducting material

Pour (copper): see Plane

Push and shove: routing a trace by pushing and shoving PCB elements surrounding the trace to create enough clearance for the new trace

Ratsnest, rat's nest: a visual representation of all unconnected nets

Resist: protected against solder(ing), see Solder mask

Routing: transforming nets into traces.

Silk screen: a non-conducting layer of graphic symbols and text, usually white, also known as component print

Short: an undesired connection between two or more pins

Solder mask: mask with openings where solder(ing) is allowed. A solder mask not only prevents short circuits, it also avoids

solder migrating away from the pad which may result in bad solder joints or pulling an SMD device out of alignment.

SMD: surface-mount device, a device using SMT

SMT: surface-mount technology, for parts with terminals that are not to be inserted in holes

Stack: neat pile of layers

Stencil: mask to apply solder paste on a PCB

Terminal: see Pin

THT: through-hole technology, for parts with terminals that are to be inserted in holes

Tombstoning: the partial or complete lifting of two-terminal SMT components during reflow

Top: the highest layer of the board stack, also known as component side

Trace: a connection between two or more pins on a PCB

Track: see Trace

Via: metalized hole connecting traces on two or more different layers

Via stitching: the use of many vias to connect a copper element on one layer to another element on another layer; often used for conducting heat or on high-frequency boards

ing is too deep, traces may disappear. If on the other hand the milling is not deep enough, shortcuts may appear. If the board is not flat, both problems may occur simultaneously. Another thing to keep in mind is that contrary to etching, milling leaves the unused copper on the board. This makes soldering harder (again, no solder mask) but can also create dangerous situations with circuits that connect to high voltages like the AC line grid because the clearances between traces are no longer respected. This can also pose problems in circuits with high-impedance inputs. It is possible to mill away the unused copper, but it will take much longer to fabricate the board.

You can manually mill a board with a Dremel-ish tool or suitable milling bits like those ball-shaped drill bits you may have seen your dentist use. Like him/her, practice and skill are required to obtain painless results.

Then there is, of course, the professional route. Pooling services that combine your design together with designs from other customers on one supersize board are widely available on the Internet. Prices and delivery terms vary wildly so it's wise to compare several providers. The costs depend mainly on board surface, number of layers, board class and production delay, and adding options will make it more expensive. Some services allow you to take options away, others don't and you may end up with features that you don't need. Pooling services in places remote to you may seem attractive but shipping delays are not always respected and packages may get lost. Some services offer a price per board including shipping, others charge setup and/or tooling costs.

Usually there is no reason to try to make the PCB as small as possible. It may be cheaper to manufacture, but doing so makes the board more difficult to route, and more complicated to integrate let alone repair.

A word on milling: sometimes non-circular milling is required to fit a part or to allow access to something. Unfortunately, milling requires yet another tool, meaning that it usually increases board costs. Some manufacturers do not add costs if the milling tool bit is the same as the one used for the board outline. Also, milling may not be possible as detailed and accurately as you would like, so, before milling away shapes, check with the manufacturer to see what is possible and what is not.

Component placement

Components should not be placed anywhere or anyway you like, not even when the circuit allows you this virtual luxury. For ease of installation, inspection and repair, components should be arranged in rows and columns, and oriented uniformly. Although for one-offs to be assembled by you this seems unnecessary, by doing so your life may become a little bit easier, as well as those of the people that assemble and repair boards for a living.

For electrical reasons components should — in general — be placed as close to each other as possible. Of course, there are many more criteria to take into account here, like current return paths, avoiding crosstalk and other unwanted inductive or capacitive couplings, etc. In short, it is important to carefully plan component placement. A power amplifier simply does not have the same requirements as a high-precision measurement device.

However, the soldering technique used to mount the components may impose yet another set of constraints. The environment too can have an impact on component placement.

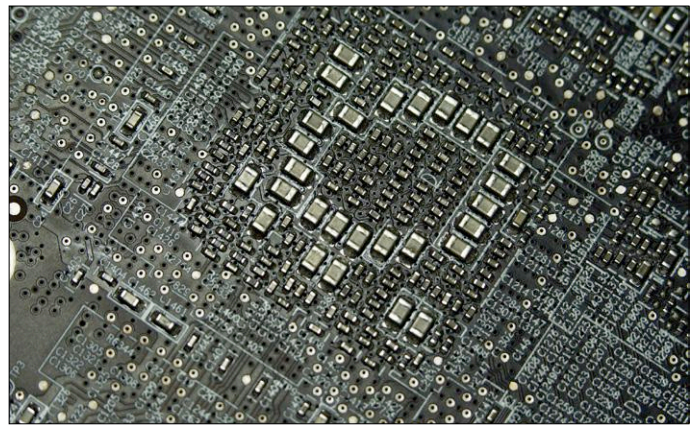


Figure 3. To provide proper decoupling for ICs with many pins, the decoupling capacitors can be placed on the other side of the board. Note how the component designators are placed outside the densely populated area in an intuitive way.

Humidity and/or dust may pollute the board, requiring larger clearances around traces and components to avoid shortcuts and leaks. This all is closely related to any norms and standards your board may be required to meet. Even though this may seem a non-issue for most DIY boards, there are good reasons why these standards exist, and reading up on them might actually teach you a thing or two.

Place THT components on the top side, also called the component side. Place SMT parts all on the same side, top or bottom; avoid placing them on both board sides as it makes manufacturing more expensive. Again, even though this may not be an issue for one-offs and small series, it is good practice to try to avoid SMDs on both sides where possible (**Figure 3**).

Boards that have SMT components on them need at least three fiducials as reference points for pick and place machines. Large ICs with many fine-pitched leads may require fiducials too to ensure correct alignment. DIY boards can, of course, do without fiducials, but why not try to make a habit of doing things the professional way? It doesn't cost anything.

Orient all polarized components such as capacitors and diodes in the same direction (unless signal integrity or other good reasons forbid you to do so). It is a great time-saver during board assembly, inspection and faultfinding. Always indicate the polarity on the silkscreen.

Beware of mirrored footprints.

Stay on the grid

Work on a grid for as long as possible. I like to place the components and do the initial routing on a 50 mil grid. When things get too dense, I will switch to a 25 mil grid. When a dense board is almost done I may switch to a 5-mil grid to squeeze the last trace segments in. Occasionally a 1-mil grid may be required to precisely position an element while respecting the design rules (DRC). Mils, mmm, it doesn't really matter which unit you choose, as long as you stick to it.

Pads

Component leads and terminals are soldered to pads on the PCB. For many parts the pads not only provide the electrical connection, they double as the mechanical mounting points for the part. It is therefore important that the pad is large enough

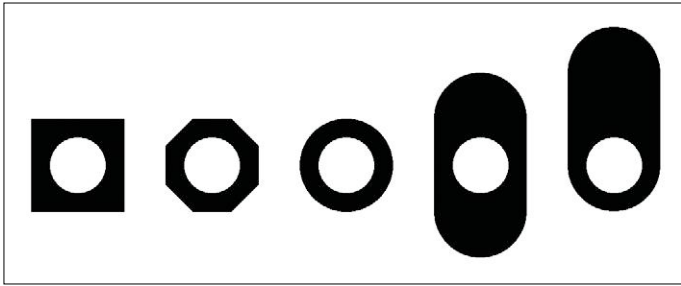


Figure 4. A selection of THT pads available in the popular Eagle PCB design tool.

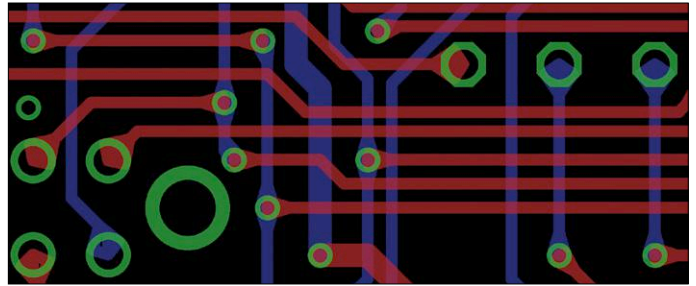


Figure 5. Tapered pads in Eagle, created with the teardrop script (ULP).

for this. Especially high or heavy parts with few terminals (e.g. large capacitors) need big pads to prevent them from accidentally being ripped off. The copper is glued to the board, and, especially when heated up (by soldering or current) delamination is a possible risk.

SMD parts usually have oblong shaped pads. Lead-free solder doesn't flow as well as traditional 60/40 solder, especially in the corners, which is why people have started to use pads with rounded corners. SMD pads must be large enough to hold enough solder paste to correctly solder a terminal. Refer to the component's datasheet for the footprint or landing pattern to use.

THT pads can have all sorts of shapes (**Figure 4**). Square pads are often used to indicate pin 1 of a part, say, a connector. Octagonal pads are popular, but round is the best shape as it maximizes the copper area (good for mechanical strength as well as heat dissipation) while simultaneously minimizing the

space required to meet clearance rules.

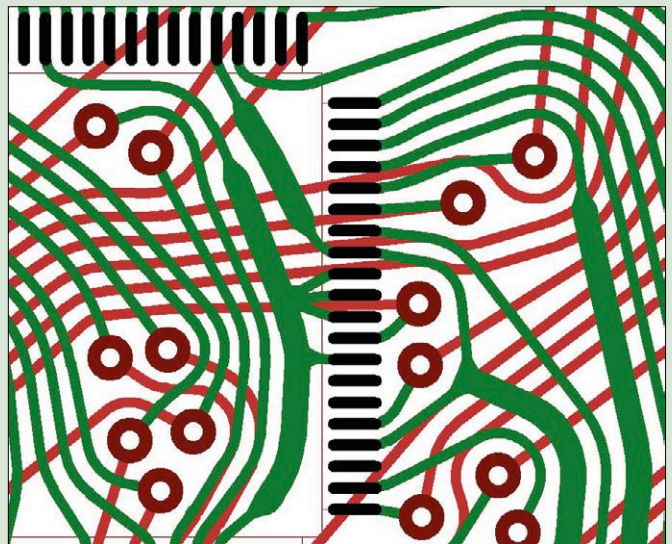
Teardrop-shaped and tapered pads — where the pad gradually merges into the connecting trace — exist too (**Figure 5**). Besides giving a slight retro touch to the board, they do provide stronger pad-trace connections. This is important for boards that have to be able to bend, like flexible PCBs. Not all PCB design tools can do this.

Then there is heat relief (**Figure 6**). When a component lead must be soldered to a very thick trace or a copper pour, the copper surrounding the pad may absorb the heat, making soldering difficult. To avoid the heat flowing away, pads can be connected with thin traces — so-called spokes or thermals, usually four — to the surrounding copper (this is where the inner annular ring makes its appearance). When doing reflow soldering (i.e. in an oven) this is less of a problem because the whole board is heated up to the same temperature. Therefore, there is usually no real reason to use heat-relief techniques

Automatic or manual routing?

CAD tool manufacturers and scientists spend and have spent many, many, many hours on developing and improving automatic routers and yet I have never been satisfied with the results of a single one of them. I also have never met anyone who was. One reason is probably that I have never had access to the best tools available, but the ones that I did try either crashed or gave up five minutes after I had left the office, never managed to route 100% of the board, if they achieved 100% then made me spend hours on cleaning it up, or were simply too complicated to set up. Because of all these reasons I prefer to route manually. (Also, I find routing a very relaxing activity.) The automatic router often manages to route up to 90% of the board or better, but hardly ever achieves 100% unless the circuit is very simple. When it gets stuck before reaching 100% this usually means that there really is no solution left to finish the board. That doesn't imply that the board can't be routed, it just means that the autorouter has blocked all possible solutions. To get out of this hole you have to undo so much that in the end it would have been quicker to have routed the board manually from the start.

As a kind of compromise, some people use the autorouter to see where routing problems may pop up, then move some components to hopefully solve these potential bottlenecks and finally route manually. Others use the auto-router only on (trivial) parts of the board and then clean up afterwards.



The Eremex TopoR 'topological router' lacks preferred routing directions to optimize space usage. (Source: Eremex)

on pads for SMT parts that are to be soldered in an oven. However, tombstoning — the partial or complete lifting of an SMT component during reflow — may occur when the thermal mass is very different on both sides of a small two-terminal device like a resistor or capacitor. Heat-relief can bypass this problem. Vias hardly ever need heat relief because, in general, they are not soldered.

Holes

Pads often have holes in them — vias have holes too. Mounting holes can be non-plated; holes in pads and vias are plated. You specify the plated-hole diameter for your vias and pads; it is the job of the PCB manufacturer to ensure that the finished hole matches this diameter. For a plated hole the actual drill size must be larger than specified because the plating material needs space. Drilling has limited accuracy and a hole may be drilled off-center (**Figure 7**). Therefore, to ensure that enough copper remains around a plated hole after drilling and etching the board, the ‘outer annular ring’ (OAR) should be wide enough. Inner annular rings (IAR) exist too. According to my dictionary, ‘annular’ means ‘shaped like a ring’, so, technically speaking, annular rings are “ring rings” (which reminds me of this great Abba song from 1973 of which I highly recommend the video on YouTube. They don’t make music like that anymore..., nor apparel, Swenglish, guitars, hairdos, etc., but I digress.)

A common mistake is to specify the wrong drill size for a component lead — either too large or too small.

There is an issue related to vias and wave soldering: if a via is unmasked, solder may flow up through it and potentially damage a component mounted over it. Closing the solder mask for vias will usually prevent this.

Traces and planes

Keep traces as short as possible; you knew that, of course. This is especially true for high-frequency signals, but low-frequency and even DC signals benefit from short traces too. Not only are short traces good for signals, they also save board space. Some fast signals may require traces (or a pair of traces) of an exact length (and impedance) which is not always the shortest connection possible.

Do not use the thinnest pen from your PCB design tools. Thin traces may cost more and are fragile. The finest detail on the board determines the board’s manufacturing class; the higher the class, the more expensive the board. Vibrations can cause micro cracks in thin traces resulting in bad or even broken connections. If the board has to be soldered by hand, or if rework is to be expected, thin traces may easily delaminate when too much heat is applied for too long. Home etching usually is not well-controlled and thin traces may get eaten away. Furthermore, a thin trace may not be capable of carrying the current that you had in mind for it. Adapt the trace width to the current that has to flow through the trace, the extra width is needed to keep the trace cool. Many online calculators are available to help you determine the best width for your trace. A thin trace carrying too much current will heat up and may eventually melt or crack; it is the heat that limits the current. Vias also must support the current. Covering traces with a thick layer of solder will increase the maximum current a trace can carry. Similarly, filling a via with solder or copper helps too. Providing openings in the solder mask will be useful here. Multiple vias “in

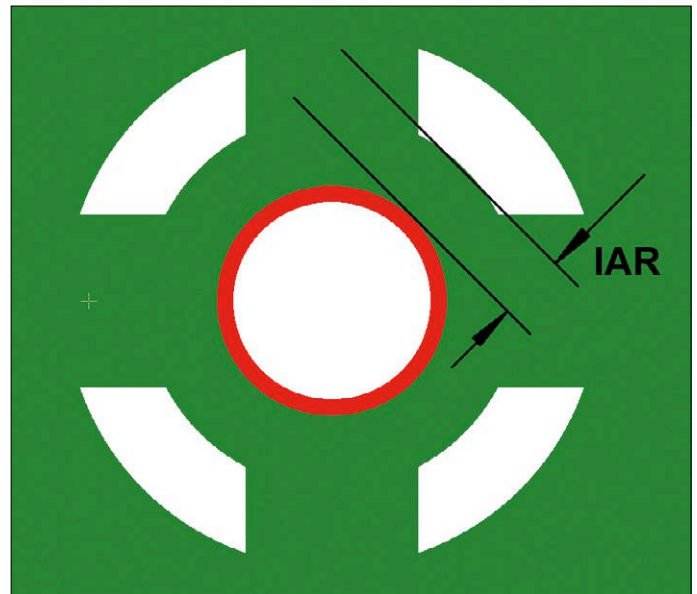


Figure 6. Heat relief prevents too much heat flowing away into the large copper pour during soldering. Here the minimum inner annular ring (IAR) may be specified even though it is much the same as the OAR. The red ring represents the plating of the hole.

parallel” are often used to improve current conduction, while limiting the risk of open circuits due to a broken via. However, one large via can be used too.

Naming nets is helpful during routing as they remind you of the signal you are working with.

Then, corners! Rounded traces are best because, contrary to angled traces, they have constant width. Changes in trace width cause impedance mismatches and reflections, but only at very high frequencies so most circuits are not affected by

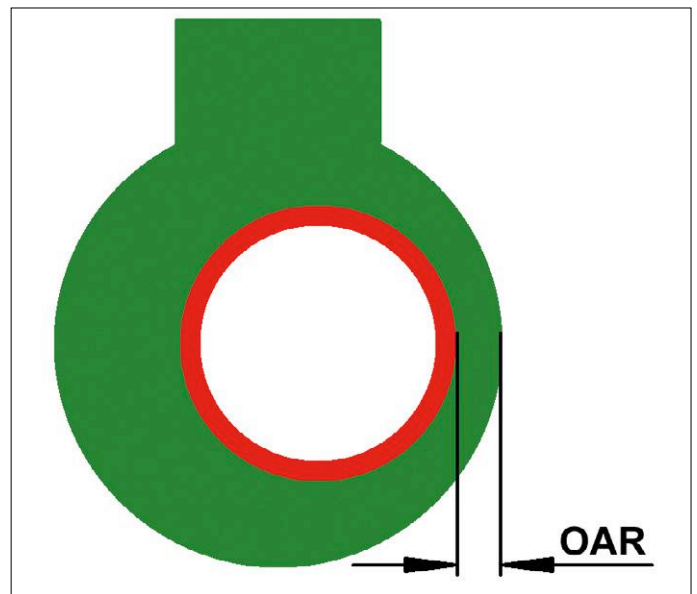


Figure 7. Due to manufacturing tolerances a hole may be drilled off-center. To allow for plating (the red ring) the hole must be drilled larger than specified. The finished hole (with plating) will have the diameter you specified. The ring that remains after drilling is the annular ring (AR). The minimum acceptable outer annular ring (OAR) shown here is specified by the design rules (DR).

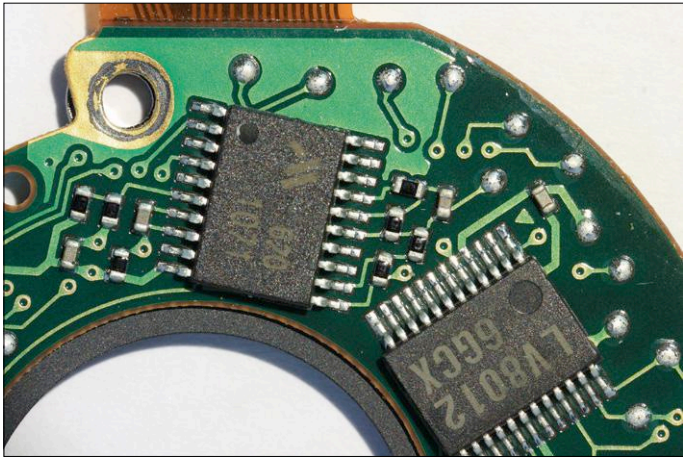


Figure 8. No component designators, only one pin-1 marking, traces with all kind of angles and teardrop vias. Once you know what you are doing it is *common practice* to be liberal about *good practice*.

these problems. Right angles are considered ugly, yet you will find many when you look closely at the vias on a board. Often a trace on one layer continues at a 90° angle on another layer, and even if it doesn't, the via itself introduces two 90° angles. Traces with 90° angles tend to be longer than those that use 45° angles. Sharp corners may also introduce delamination problems, acid pockets, or on the other hand, under etching. Sometimes it is impossible to avoid a sharp edge and in most cases that is fine. Personally I tend to stick to 45° angles whenever I can.

Using a power plane or copper pours (usual for GND) can save a lot of work and is good from an EMC perspective. However, be careful when using multiple supply planes, especially close to board edges and mounting holes where metal objects and fixing screws may create unintended short-circuits between planes. Always run an electrical rules check (ERC) when done to ensure that all short and open circuits have been corrected. When you use a plane or pour for a net, do not route that net but let the copper pour take care of it. Beware of very thin, sort of accidental, connections between plane segments that may get etched away and break the continuity of the plane. Such planes are unsuited for home etching. To prevent this from happening, start with a high copper pour clearance and see if the copper flows everywhere. Move traces and vias to improve copper flow. When you are satisfied with the pour you may decrease the clearance.

Note that planes have an influence on the copper distribution of the board. When the copper is unevenly distributed etching may become uneven, and can result in uneven copper thickness distribution making the PCB bend at extreme temperatures (as in automotive applications).

Use design rules

Use the design rules checks (DRCs) and electrical rules checks (ERC) to make sure all PCB elements respect them. Good design rules will help avoid traces from overlapping or coming too close to the board's edge, help you find illegal drill sizes and clearances and much more. Use them! Also make sure that when you think the board is ready, all nets are connected indeed. Do not leave warnings and errors even if they are acceptable for you because the person that inherits the design — or yourself

when you have another look at it six months later — may get confused by them. If you must accept a warning, document the reason why. If possible change the design rule that gives you trouble to make it go away, but make sure the new rule is acceptable for your application as well as for your board manufacturer.

Markings and component print

Try to provide designators for all component outlines and make sure they remain visible after the components are mounted. I like to orient designators the same way as the components, in rows and columns (see also Figure 3). Do indicate pin 1 on all connectors, headers, and any other component where pin 1 is not readily identifiable. Also clearly mark the polarity of all polarized components such as capacitors and diodes (**Figure 8**). Avoid print under SMT parts, especially two-terminal ones, as this makes their "seat" wobbly and may result in tombstoning. Keep text legible and use labels where useful to guide the user. Note that the openings in the solder mask are usually slightly larger than the pads, and any silkscreen print crossing these openings is cut away so make sure text does not overlap them. Put legible text on all layers to prevent accidental mirroring of a layer. Label or number layers to ensure that they will be stacked in the right order.

Give the board a unique name or number and don't forget to mark its revision.

If the board's bottom side can have print too, use it.

DIY boards in general do not have a silkscreen, but that doesn't mean that you should forget about text and polarity markings; you can write on the copper layers too. So, where possible place marks and text while keeping an eye on their size to avoid that they will be lost during etching.

Testing

Provide easily accessible test points. If possible make the circuit produce useful test signals. Component terminals should not be used as test pads as a test probe pushed against a terminal may temporarily "repair" a bad solder joint, feigning a "good" connection that's bad most times. Vias can be useful for testing, but only when they were left open in the solder mask.

Is that all?

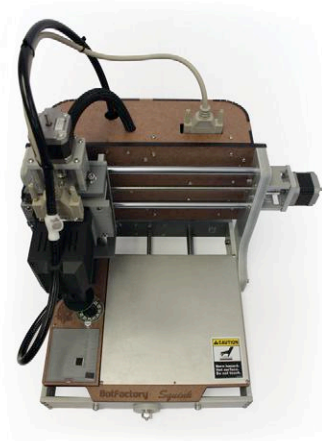
After reading this long article you may think that you know a lot about designing PCBs, but in reality we only scratched the surface. We did not discuss, to name a few, current return paths, multiple ground planes, heat management during soldering and heat management when operating, EMC compliance, handling high-speed signals, designing footprints, etc. PCB design is a vast theme combining chemistry, physics, electronics, mechanics and automation. It is rather amazing to discover that many aspects of PCB design never really have been studied in detail and are simply based on common sense and assumptions. On the Internet you can find many discussions on PCB design details, so, when in doubt, ask around.

Thanks to Malte Fischer for his useful suggestions. ◀

(160397)

Web Link

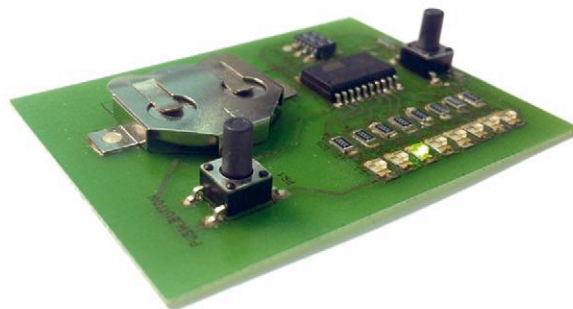
[1] www.elektormagazine.com/160397



We saved you some space

And time, and money too. BotFactory Squink is a desktop PCB Factory, capable of printing a 2-layer PCB, dispensing conductive glue or paste and pick-and-placing the components. All the elements of a PCB line, condensed into a single, portable package.

Squink is ideal in the hands of researchers, educators and businesses both small and large. When it comes to prototyping, BotFactory wants to save you the inevitable wait or the unbelievable expense of sending out orders for your boards.

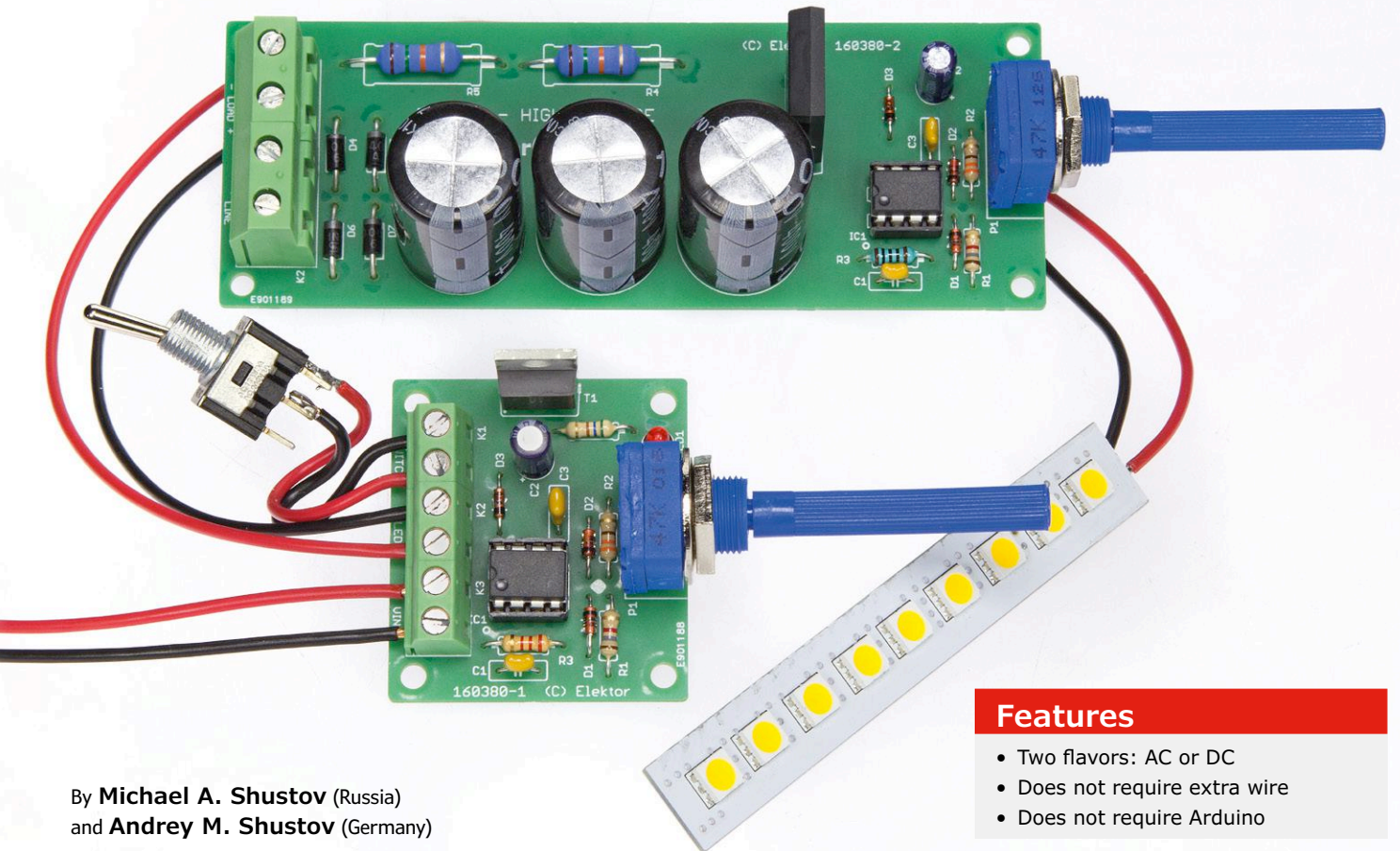


We can't save you from your curiosity, so go to BotFactory.co to learn more.



2-Terminal Dimmer

controls LEDs, lamps and heaters



By **Michael A. Shustov** (Russia)
and **Andrey M. Shustov** (Germany)

Features

- Two flavors: AC or DC
- Does not require extra wire
- Does not require Arduino

In the past, rheostats (today also called potentiometers) were connected in series with a load to control the current flowing through the load. Although simple, this technique has some disadvantages.

First of all, the current can only be controlled when the load is active. Secondly, the heating of the rheostat results in significant losses. Furthermore, controlling the intensity of a light flux emitted by light-emitting diodes (LEDs) with a rheostat is almost impossible owing to the nonlinear current-voltage characteristics of LEDs. Pulsewidth modulation (PWM) is a better and much more efficient way to smoothly change the intensity of heating and lighting devices (incandescent

lamps and LEDs), or to control the speed of a motor.

Dimmers controlling the intensity of light sources are usually connected in parallel with the power supply; they are three-terminal devices. Two terminals connect to the power supply while the third connects to the load. Such dimmers

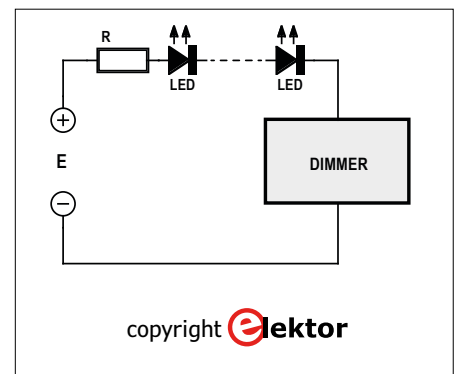


Figure 1. The dimmer is in series with the LED(s) and requires only two wires.

copyright **e**lektor

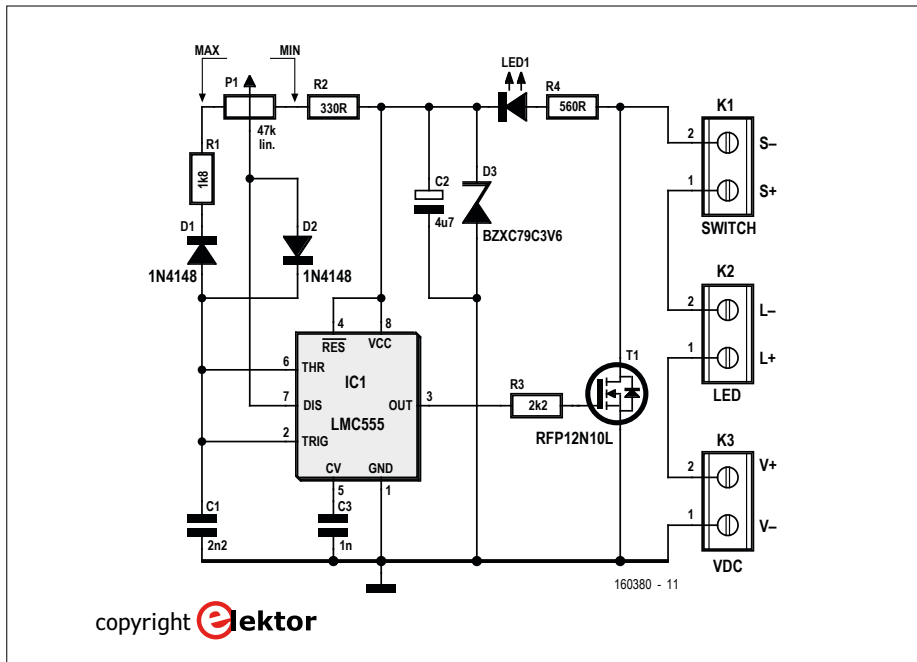


Figure 2. The DC version of the series dimmer.

cannot be used in place of conventional double-pole switches, since they require an additional third power wire for proper connection. To overcome this problem we propose a two-terminal dimmer connected in series with the load (LED, lamp or a heating element) instead of a switch, see **Figure 1**. Contrary to a switch, the dimmer allows adjusting the load current from almost 0 to about 97% of its maximum value. The dimmer is powered through the load when its switching element is disconnected and the current through the load is minimal. An inconvenience of both two-terminal and three-terminal dimmers is their residual current consumption, but it is insignificant compared to the current when the load is powered. The dimmers described below can be used with household lighting devices, thermostats, and for instance back-up or secondary lighting.

Two-terminal DC-powered dimmer

At the heart of the dimmer (see **Figure 2**) is the good old 555 timer IC in its low-power CMOS guise called LMC555CN. It is wired as a pulse generator with variable duty cycle, a well-known circuit, nothing new under the sun. The output of IC1 drives power MOSFET T1 which in turn switches on and off the load (the LED) as dictated by the PWM signal. The PWM signal has a frequency of approx-

imately 6 kHz, i.e. way too fast for the human eye to notice. The pulse/pause ratio or duty cycle is adjusted with potentiometer P1; the minimum and maximum values for the pulse width are determined by resistors R1 and R2.

Up to here the dimmer is a standard three-terminal shunt regulator. The trick to get rid of one terminal is to use the load's supply to power the circuit. This is achieved by charging C2 through R4 and LED1 when T1 is not conducting. Zener diode D3 limits the voltage to 3.6 volts. LED1 serves as a power-on indicator and also makes it easier to find the dimmer in the dark.

Transistor T1 can withstand up to 100 V and conduct 12 A, but not at the same time (the PCB only handles up to about 2.5 A anyway). According to its data-sheet it can dissipate up to 60 watts, and it should be mounted on a suitable heatsink when it has to handle more than 1 watt. R4 determines the maximum supply voltage, 20 V here.

With the values given in the schematic we measured a quiescent current of 3.5 mA (i.e. when the duty cycle is 0%). That's enough current to make certain LEDs light up noticeably. Reducing the value of R2 to 75-100 ohms may improve the situation. In addition, increasing the value of R5 will reduce the minimal current too. A switch can be connected to K1 to completely turn off the dimmer.

PROJECT INFORMATION

LED

lighting PWM dimmer

Danger! 555

entry level

→ intermediate level

expert level

2 hours approx.

Soldering iron, common sense

£12 / €15 / \$17 approx.

AC Line-powered dimmer

It is, of course, possible to adapt the dimmer for use with an AC power supply like the 230 VAC line voltage available in households in many countries all over the world (115 VAC will work too.) To achieve this feat two hurdles have to be taken:

- Finding a suitable power transistor capable of handling such high voltages;
- converting the AC voltage to a DC voltage.

The second point requires some attention. Simply rectifying the AC line voltage is not good enough, as heavy loads like incandescent light bulbs will flicker, making filtering mandatory. Furthermore, the rectified line voltage is about 325 V (163 V for 115 VAC line voltage), which is too high for standard household light bulbs (several identical lower-voltage light bulbs connected in series can be used, however). Because of these practical limitations, what follows is mainly to illustrate our point and is not recommended for use in real life.

Figure 3 shows the 230-VAC version of the dimmer. The most noticeable change is the addition of the rectifier (D4-D7) and the supply filter made up with C4, C5 and C6. We also added a skull & bones symbol to make clear that potentially lethal things are going on here. The filter capacitor is made with three

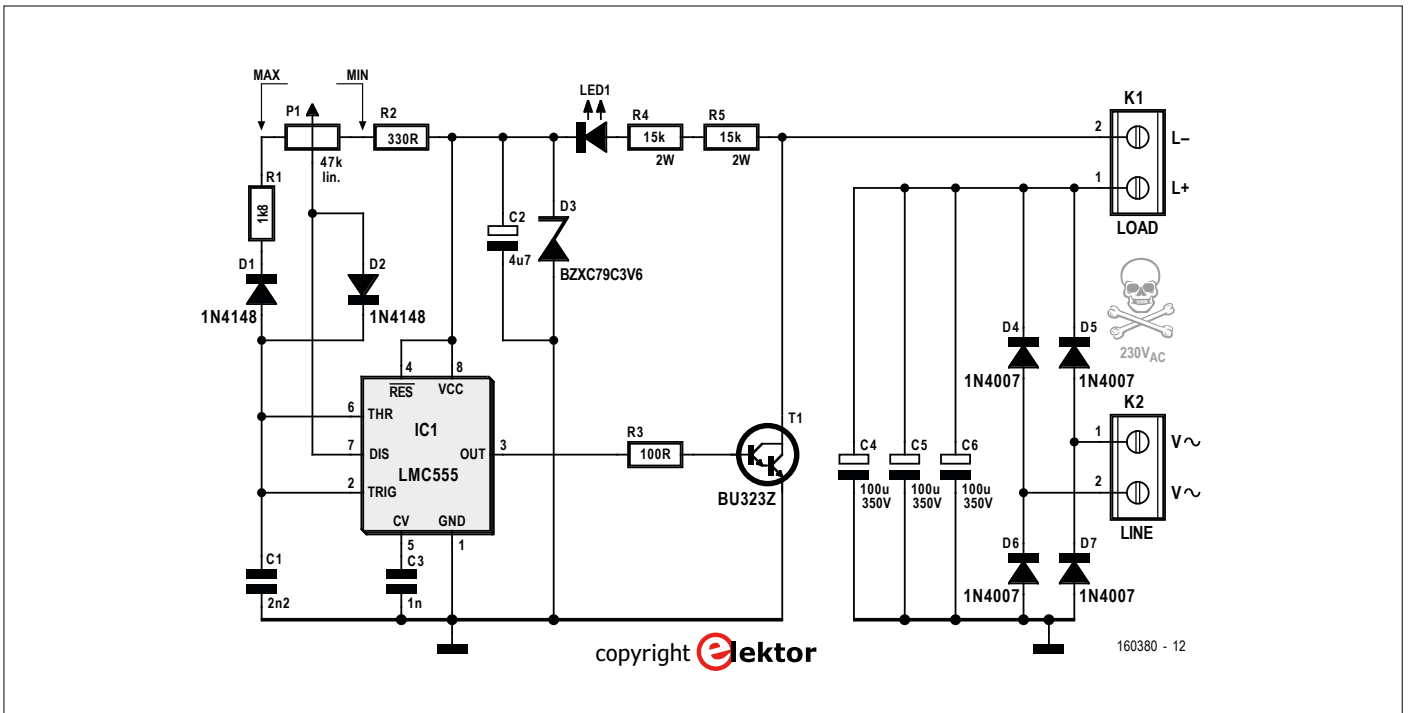


Figure 3. This dimmer circuit is dangerous and may kill you. Do not look at it too long.

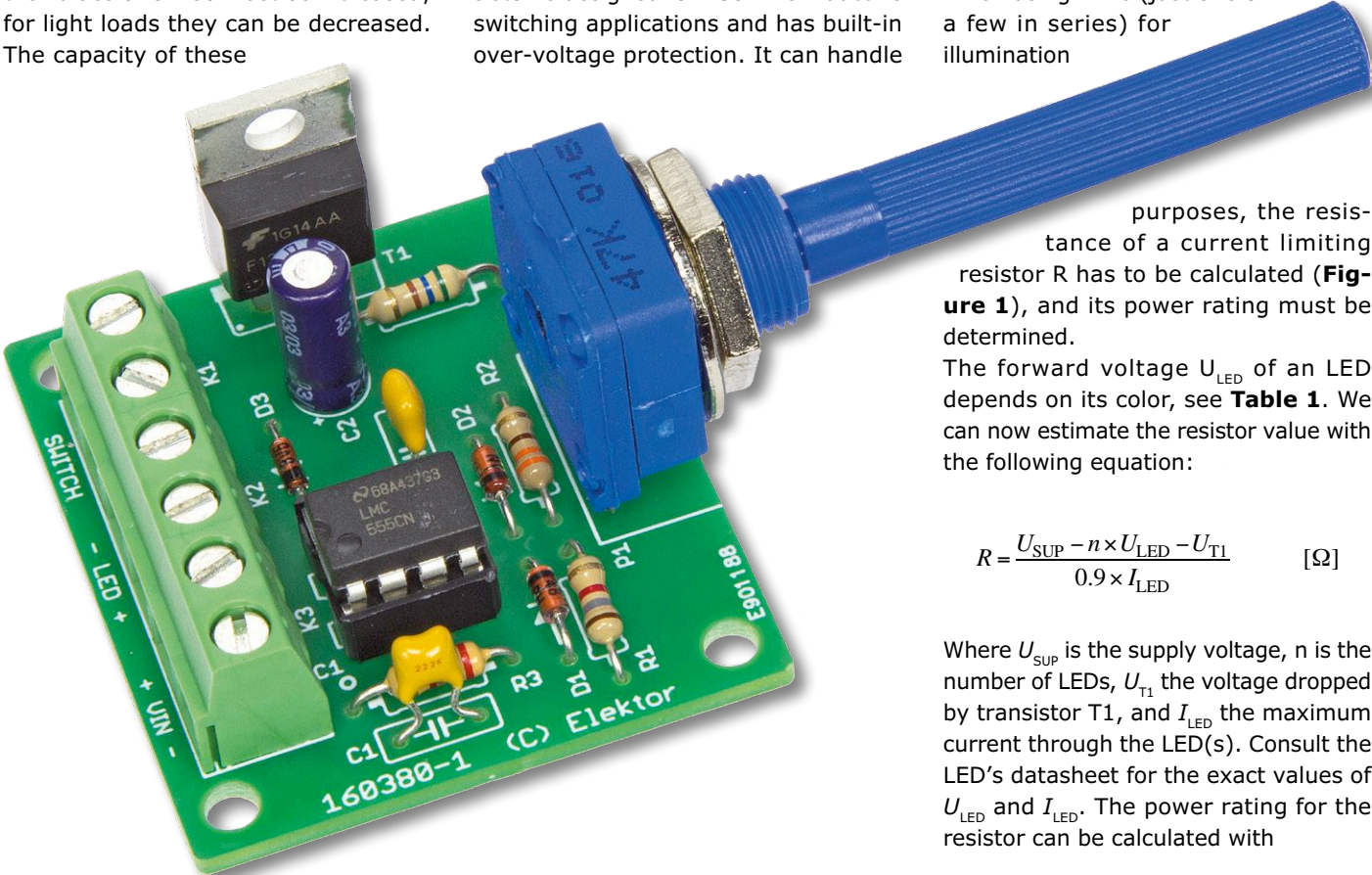
separate devices because one big one is difficult to find and rather expensive. With three times 100 µF (350-V types minimum) and a 100-W light bulb the voltage ripple at the output will not exceed 3% (approx.). This is invisible to the human eye. For heavier loads the values of C4-C6 must be increased; for light loads they can be decreased. The capacity of these

capacitors is not critical when a heating element is used as a load. Besides the clearly visible schematic changes there are also a few more subtle changes. Transistor T1 for once is now a BU323Z Darlington power transistor instead of a MOSFET. This transistor is designed for 230-VAC inductive switching applications and has built-in over-voltage protection. It can handle

up to 150 W (with a suitable heatsink!). If you are tempted to replace this transistor by something “sturdier”, keep in mind that the rectifier diodes D4-D7 are specified up to 1 A and 3 W.

Let’s do some math

When using LEDs (just one or a few in series) for illumination



purposes, the resistance of a current limiting resistor R has to be calculated (Figure 1), and its power rating must be determined.

The forward voltage U_{LED} of an LED depends on its color, see Table 1. We can now estimate the resistor value with the following equation:

$$R = \frac{U_{SUP} - n \times U_{LED} - U_{T1}}{0.9 \times I_{LED}} \quad [\Omega]$$

Where U_{SUP} is the supply voltage, n is the number of LEDs, U_{T1} the voltage dropped by transistor T1, and I_{LED} the maximum current through the LED(s). Consult the LED’s datasheet for the exact values of U_{LED} and I_{LED} . The power rating for the resistor can be calculated with

$$P_R = (I_{LED})^2 \times R \quad [W]$$

Examples

- (Figure 2) Assume $n = 1$, $U_{LED} = 3 V$, $I_{LED} = 0.05 A$, $U_T = 0.04 V$, $U_{SUP} = 20 V$. Then $R = 377 \Omega$, or, rounded off to the nearest higher standard value, 390Ω . The resistor should be able to absorb $(0.05)^2 \times 390 = 0.975 W$, or, rounded off to a common value, $1 W$.
- (Figure 2) Assume $U_{LED} = 2 V$, $I_{LED} = 0.02 A$, $U_T = 0.04 V$, $U_{SUP} = 16 V$. If we limit $n \times U_{LED}$ to 80% of U_{SUP} , then $n = 6$. In this case R becomes 220Ω , and its power rating should be better than $88 mW$ ($0.125 W$ would be a good standard value).
- (Figure 3) Assume $U_{LED} = 2.5 V$, $I_{LED} = 0.05 A$, $U_T = 3 V$, $U_{SUP} = 325 V$. If we limit $n \times U_{LED}$ to 80% of U_{SUP} , then $n = 104$ and $R = 1,378 \Omega$. The nearest standard value would be $1.5 k\Omega$. Its power rating should be better than $3.75 W$. The required resistor can be realized by connecting two $2 W$, $3 k\Omega$ resistors in parallel. With

Table 1. Forward voltages related to the color of the light emitted by an LED.

Color	Wavelength [nm]	U_{LED} [V]
Infrared	≥ 760	≤ 1.6
Red	610 – 760	1.6 – 2.0
Orange	590 – 610	2.0 – 2.1
Yellow	570 – 590	2.1 – 2.2
Green	500 – 570	2.2 – 2.5
Blue	450 – 500	2.5 – 2.7
Violet	400 – 450	2.7 – 3.1
Ultraviolet	≤ 400	≥ 3.1
White	-	3 – 3.7

these numbers the current through the LEDs becomes $(U_{SUP} - n \times U_{LED} - U_T) / 1500 = 0.041 A$, meaning that T1 has to dissipate $U_T \times I_{LED} = 0.125 W$. It should be able to do this without a heatsink.

by the use of the circuits presented in this article. ◀


(160380)

Web Link

[1] www.elektormagazine.com/160380

Final words

We can't repeat it enough times: the circuit from Figure 3 is dangerous; it carries lethal voltages and may kill you or someone else when used without having taken proper precautions first. Do not build this circuit. Do not use it either, even if you didn't build it. The authors and Elektor take no responsibility or liability, so far as legally possible, for any damages caused



FROM THE STORE

→ 160380-1
PCB for DC-version



COMPONENT LIST

Resistors

- R1 = 1.8k Ω
- R2 = 330 Ω
- R3 = 2.2k Ω
- R4 = 560 Ω
- P1 = 47k Ω , potentiometer, linear

Capacitors

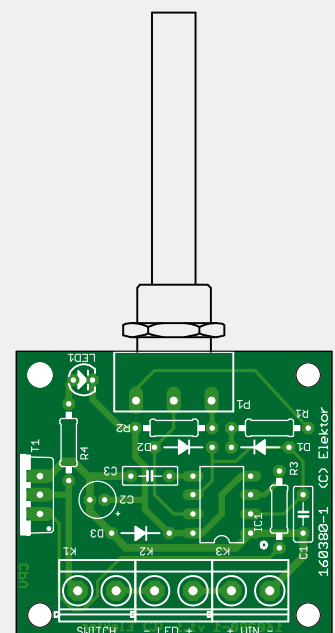
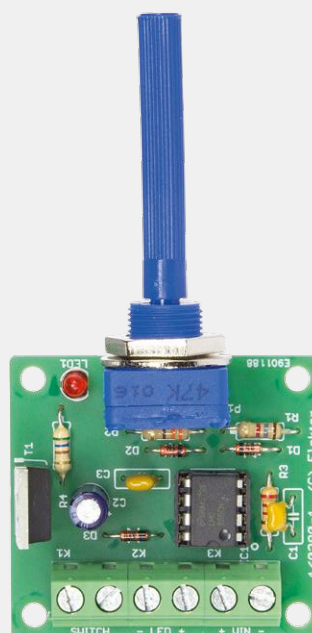
- C1 = 2.2nF, 0.2" pitch
- C2 = 4.7 μ F, 2mm pitch
- C3 = 1nF, 0.2" pitch

Semiconductors

- D1,D2 = 1N4148
- D3 = BZX79C3V6
- IC1 = LMC555CN
- LED1 = red, 3mm
- T1 = RFP12N10L

Miscellaneous

- K1,K2,K3 = 2-way PCB screw terminal block, 0.2" pitch
- PCB # 160380-1





RF Step Attenuator

Adjustable attenuation gives precise signal levels

By Alfred Rosenkränzer (Germany)

When dealing with radio-frequency signals it often comes in handy to have an easy way to attenuate a signal level in discrete steps. For example, you might want to reduce the output of an RF generator to a lower level than the output control on the generator allows, and there are many other applications of attenuators in the adjustment and repair of receivers and other radio equipment. Unfortunately professional attenuators are rather pricey, but they do not contain any expensive electronics and so a DIY approach is an eminently practical alternative.

Sometimes it is necessary to attenuate an RF signal by a known amount. If the signal comes from an RF source whose level cannot easily be adjusted, then

there is no real alternative to an external attenuator, which is normally simply inserted into the RF output connection. Such an arrangement is useful, for

example, when testing the sensitivity or small-signal behavior of a receiver: just insert the attenuator in the antenna lead and increase the degree of attenuation



Features

- Switched RF attenuator with six stages, 0 dB to 31 dB attenuation in 1 dB steps

to the point where reception starts to fail or where whatever phenomenon is being investigated, perhaps noise or AFC artifacts, starts to raise its head. Many other applications of attenuators will be immediately obvious to any RF engineer. The familiar names in test equipment manufacture produce good-quality attenuators (see **Figure 1**) but unfortunately at a price that not all experimenters will be comfortable shelling out. But an attenuator basically consists only of an enclosure, connectors, switches and a few resistors. With a little care it is perfectly possible to build a good attenuator for the common frequency bands yourself for a much more reasonable price.

Attenuator circuit

There are two different but equivalent circuit configurations used to attenuate signals while maintaining impedance matching (in other words, the input and output

impedances of the circuit are the same, but the signal level is not). These are called a T-type attenuator and a pi-type attenuator. Each configuration consists of three resistors (see **Figure 2**). For this project I selected the pi-type attenuator as it makes the printed circuit board

layout more convenient. When working with RF a printed circuit board is preferable to hand wiring, as it ensures that all stages can be made geometrically identical and helps keep parasitic inductances and capacitances under control.

Figure 3 shows the resulting circuit of a



Figure 1. Example of a VHF attenuator by HP, long since out of production.

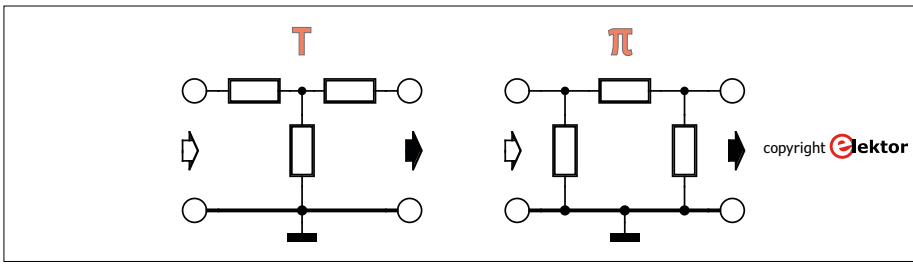


Figure 2. T-type and pi-type attenuator networks.

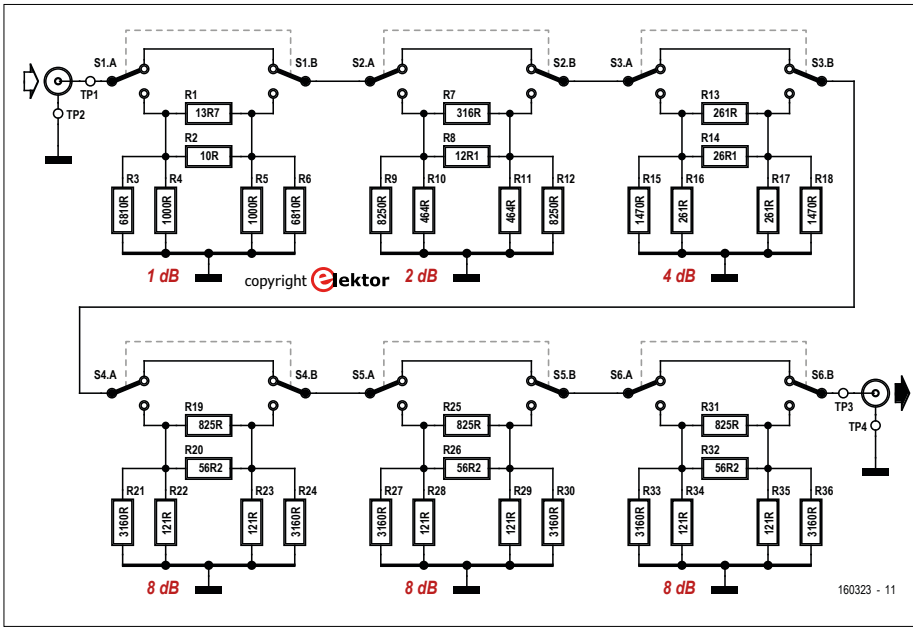


Figure 3. Complete circuit of the six-stage attenuator.

six-stage attenuator. It simply consists of six cascaded pi-type attenuator stages each of which can be optionally bypassed using a double-pole switch. A point to note is that each resistor in each pi network has been replaced by a parallel combination of two resistors. This allows the calculated ideal value of each resistor to

be realized accurately despite only using standard E-series values. In total, then, the circuit comprises six switches and (up to) $6 \times 3 \times 2 = 36$ resistors. The desired attenuation of each stage is achieved by choosing appropriate values for the resistors. In theory any attenuation between 0 dB and ∞ dB can be

realized; however in practice the maximum attenuation is limited to about 10 dB, as beyond that parasitic effects in the components, printed circuit board and switches (which are not specially designed for RF applications) come into play. The smallest degree of attenuation used in the circuit is 1 dB, although fractions of a decibel could of course be implemented.

If a switch is in its upper position, its stage is bypassed and the signal is not attenuated. In the lower position, the signal passes through the attenuator stage. The attenuation levels of the individual stages can be chosen independently, as the input and output impedances of all the stages are the same. In my prototype I built attenuators of 1 dB, 2 dB, 4 dB, 8 dB, 8 dB and 8 dB. Together these allow RF attenuation in steps of 1 dB (about a 10 % reduction in amplitude) from 0 dB to 31 dB, which makes sense for the typical uses to which I put the device. Other combinations are of course possible: for example six 10 dB stages would allow attenuation from 0 dB to 60 dB in steps of 10 dB, or stages of 0.5 dB, 1 dB, 2 dB, 4 dB, 8 dB and 8 dB would give attenuation from 0 dB to 23.5 dB in steps of 0.5 dB.

Calculations

In order to calculate the required resistances I used the AADE software, which can be downloaded at [1]. The main use of this program is in creating more complex designs such as filters, but it is perfectly capable of helping to design attenuators and saves a lot of tedious calculation.

In the menu Design -> Attenuator Pad first select the 'pi' topology and in the next menu select the source impedance. In principle any value might be used here, but these days the conventional values are 50 Ω for test equipment and 75 Ω for antennas and analog video signals. The printed circuit board is designed for an impedance of 50 Ω . If required, the trace widths and board thickness can be modified for use in a 75 Ω environment. Programs to help with this design can be found on the Internet: if you are thinking of making such modifications and understand the issues involved, you may find the tool linked to at [2] worth a look. The desired attenuation value is entered in the next menu and then the program shows the circuit with the required resistor values. Unfortunately the values

COMPONENT LIST

Resistors
36 pcs SMD case 0603 or 0805; see text for values and tolerances

Miscellaneous
S1-S6 = DPDT changeover switch, PCB mount (e.g. Reichelt TL 46 PO)
Cast aluminum enclosure, Hammond type 1550Z102
RF connectors as required (e.g. panel mount BNC sockets)

Figure 4. RF layout techniques are used on the attenuator circuit board.

PCB: see text

almost never coincide with E-series preferred values, and even if we allow ourselves to use E96 values it is not always possible to find a good solution. Instead it is possible to come up with pairs of resistors from the readily-available E24 series whose parallel combination hits the computed ideal value sufficiently closely, using either mental arithmetic, a pocket calculator, an Excel spreadsheet or a short program. An even easier approach is to use one of the online tools that are available: these do all the hard work and show possible combinations along with the percentage error in the result. A good choice is the tool found at [3]. For this application errors of up to 1 % are acceptable.

You might wonder why the circuit uses parallel resistor combinations rather than series combinations: the reason is that it simplifies the design of the circuit board and improves its RF performance.

It's all down to the board

The board I laid out for this circuit is shown in **Figure 4**, and the design files (in Eagle format) are available for free download from the Elektor web page accompanying this article [4]. The switches are in the middle, and immediately below each are the six resistors corresponding to it. There are three vias connecting the network to the ground plane on the other side of the board. The trace widths, when used with an FR4 substrate 1.0 mm thick (this dimension is important!), are chosen to yield a characteristic impedance of 50 Ω . The resistors are intended to be 0603 SMD types, but their pads are sufficiently large that it is easy to fit 0805 components instead. The power handling capability of the attenuator depends directly on the power rating of the resistors used: the design is not intended for use in high-power applications, and it is best to keep power levels to below 100 mW. The layout also includes, above the main circuitry, a test structure and provision for two SMA connectors to allow measurement of the characteristic impedance of a trace. The connectors are optional and only required for this measurement. The board is designed to fit neatly into a Hammond cast aluminum enclosure (see the component list). Input and output are on RF sockets: BNC connectors are conventionally used in test equipment applications. Populat-

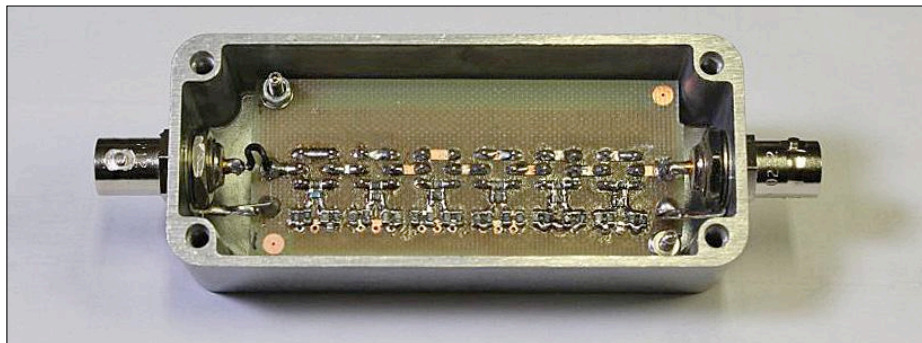


Figure 5. The circuit board fitted in the Hammond enclosure.



Figure 6. The completed prototype.

ing the circuit board should not present any special difficulties, although soldering 0603 SMD resistors calls for good tweezers and a steady hand. After assembling the board it should be thoroughly tested by enabling each attenuation stage individually in turn (with the other five switches set bypass their stages) and then measuring the input and output impedances of the unit as a whole. When measuring the impedance at one port it is of course necessary to terminate the other port in 50 Ω (for example, using two 100 Ω resistors in parallel). If the measured impedance is within 1 % of 50 Ω , then you can be fairly sure that you have not accidentally got two resistors the wrong way around. The final test is to make sure that the attenuation introduced by each stage is correct. This can be done at DC: use a power supply (at no more than 5 V) and a 50 Ω series resistor to feed the input of the device and terminate the output with 50 Ω . Measuring the ratio between the voltage at the input terminals of the device and the voltage at the output terminals of the device will give the actual attenuation.

Figure 5 shows the board fitted in



Figure 7. Screenshot showing the frequency response at minimum (0 dB) and maximum (31 dB) attenuation.

the Hammond enclosure and **Figure 6** the rather handsome (even if I say so myself) completed prototype. And it's not just a pretty face: considering the simplicity of the design its performance is more than acceptable. **Figure 7** shows the frequency response at attenuations of 0 dB and 31 dB: as can be seen, the attenuator performs well up to around 200 MHz. ◀

(160323)

Internet Links

- [1] <http://w1hue.org/filter.html>
- [2] www.eeweb.com/toolbox/microstrip-impedance
- [3] www.qsl.net/in3otd/parallr.html
- [4] www.elektormagazine.com/160323

About the Author

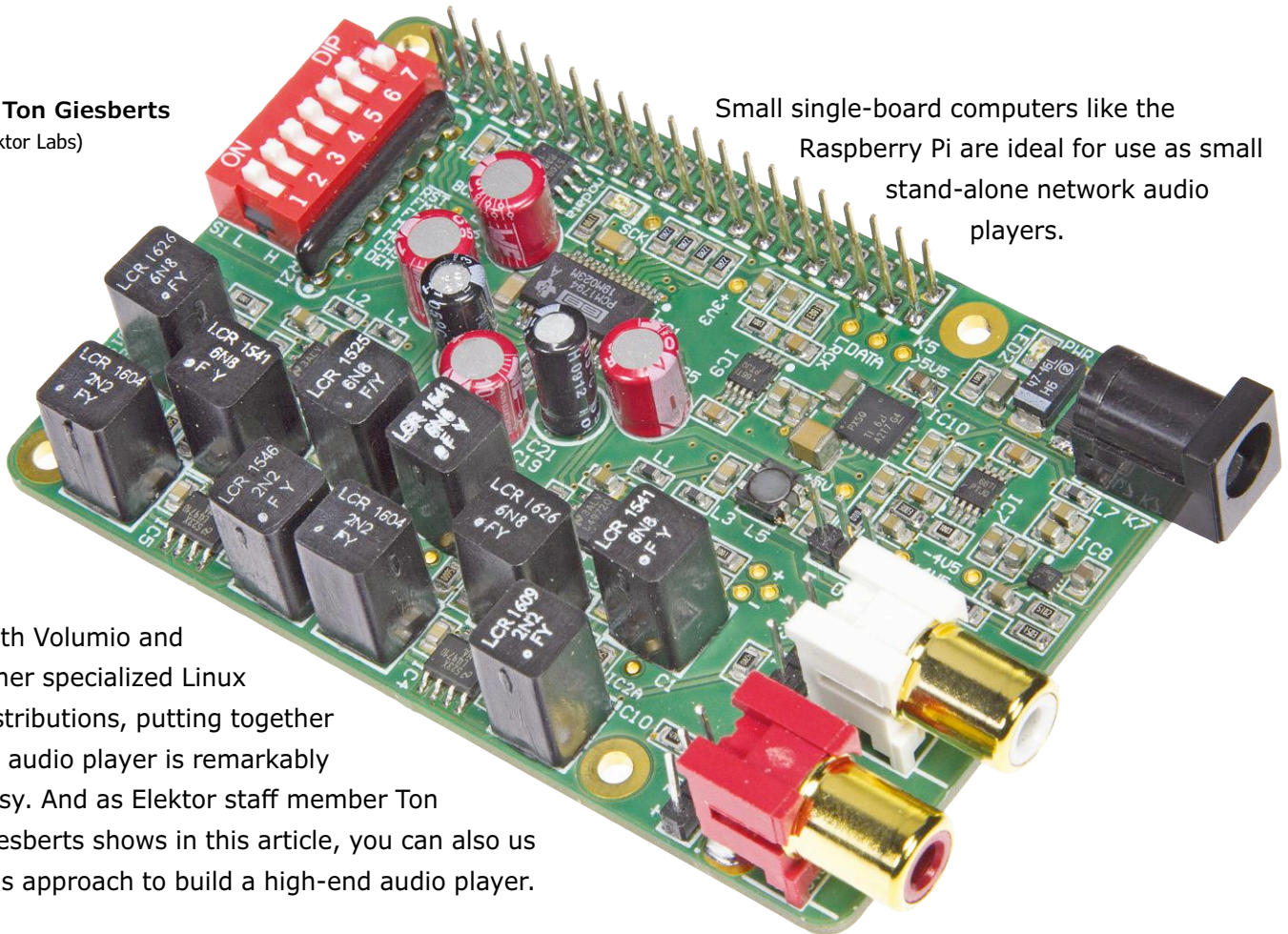
Alfred Rosenkränzer has worked for over 30 years as a design engineer, initially in the field of professional television equipment. Since the end of the 1990s he has worked on high-speed digital and analog circuits for IC testing.

Audio DAC for Raspberry Pi

A network audio player with Volumio

By **Ton Giesberts**
(Elektor Labs)

Small single-board computers like the Raspberry Pi are ideal for use as small stand-alone network audio players.



With Volumio and other specialized Linux distributions, putting together an audio player is remarkably easy. And as Elektor staff member Ton Giesberts shows in this article, you can also use this approach to build a high-end audio player.

If you are looking for a small stand-alone network audio player, preferably with a touchscreen user interface, you don't have a lot of options. Usually you end up with a largish amplifier from one of the big brand names, with a corresponding price tag. And most of them do not have a touchscreen. If what you want is something compact and portable, these devices do not fit the profile. That means your only real option is to make your own. Apps based on the Raspberry Pi are a good starting point, and DACs with fairly good specs are now available for the RPi platform. However, at Elektor we are not satisfied with "fairly good" – if we are going to develop a

project for this, we might as well design a good high-end DAC for the RPi using components with top-notch specs.

The project at a glance

Digital to analog converter (DAC) ICs from Burr Brown (now part of Texas Instruments) have always been (and still are) about the best you can buy, and we have used them previously in various high-end DAC projects. For this project we opted for one of the top of the line devices from Burr Brown: the PCM1794A. This is a 24-bit DAC which can handle sampling rates up to 200 kHz and has an integrated 8× oversampling filter. This IC comes in a 28-pin SSOP

package, and it has outstanding specs in all respects, including a dynamic range of 127 dB ($2 V_{RMS}$, stereo) to 132 dB ($9 V_{RMS}$, mono), a THD of 0.0004%, and differential current outputs ($7.8 mA_{pp}$). The PCM1794A accepts all known data formats: standard, left justified, and I²S (which is important for our project). The digital portion of the IC operates at 3.3 V, but the digital inputs are also compatible with 5 V signals. For a complete overview of the specifications, see the device data sheet [1].
In order to convert the differential current outputs into single-ended voltages, we need a current to voltage converter for each channel, followed by a combined

filter and gain stage that converts the differential voltage output into a single-ended voltage output.

Based on this knowledge, we can draw the block diagram shown in **Figure 1**. From that we can derive the following shopping list: on the hardware side a Raspberry Pi 3 (version 2 is also usable) and the previously mentioned audio DAC, along with a Waveshare 3.5" touchscreen display for the Raspberry Pi to provide the user interface, and on the software side Raspbian for the RPi, Volumio or something similar (such as Mood Player), and the driver for the Waveshare display.

Power supply considerations

The digital to analog converter IC1 (see the detailed schematic diagram in **Figure 2**) needs two different supply voltages: +3.3 V for the digital portion and +5 V for the analog

converters and analog output filters, we use an on-board inverting voltage regulator to derive it from the +5 V rail. If you take a closer look at the circuit diagram, you can see that the analog supply voltage is actually +5.2 V instead of +5 V. We intentionally chose this somewhat higher voltage, which is still within the safe range for the DAC IC, to ensure a maximum undistorted signal level of 1 V from the output filter. You often see an output voltage of 2 V, but for that you have to use rail-to-rail opamps or a higher supply voltage, and most rail-to-rail opamps cannot match the outstanding specifications of special high-end audio opamps.

We chose a TPS7A4700 ultra low-noise, low-dropout voltage regulator to provide the +5.2 V main supply voltage. It can supply a maximum current of 1 A with a voltage drop of just 307 mV. The output

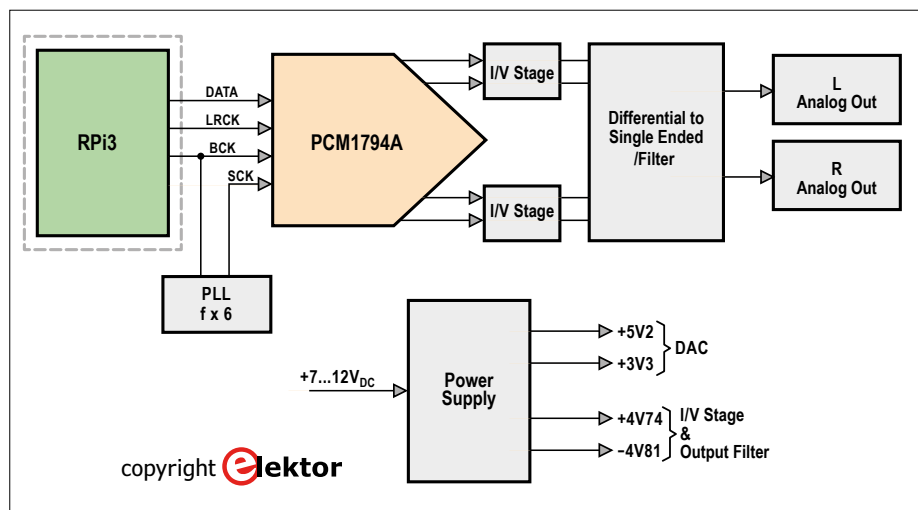


Figure 1. Block diagram of the audio DAC.

portion. In theory we could use the supply voltages available on the GPIO expansion connector of the Raspberry Pi board, but they are so noisy that there would be nothing left of our high-end aspirations. That means we have to provide an independent source of +3.3 V and +5 V. Of course, it would also be possible to power the Raspberry Pi from the DAC board, but that would have the same detrimental effect on the ultimate signal quality. A drawback of this totally separated approach is that we will need two AC adapters to power the overall device. To avoid the need for yet a third AC adapter to provide the negative supply voltage for the current to voltage

voltage can be set by grounding specific pins which are connected to internal resistors. Each pin has an associated voltage, and the output voltage is equal to the sum of the voltages of the individual pins. The minimum output voltage is the same as the reference voltage of 1.4 V. To obtain an output voltage of +5.2 V, we tie pins 6, 10 and 11 to ground, so the output voltage is given by the formula:

$$V_{out} = 1.4V + 3.2V + 0.4V + 0.2V = 5.2V$$

Other voltages

The positive supply voltage for the analog output portion is set to +4.74 V, which is

PROJECT INFORMATION

audio

network player

DAC

Raspberry Pi

touchscreen

entry level

intermediate level

expert level

5 hours approx.

Raspberry Pi 2 or 3, com-
puter with Putty (Windows)
or Linux, two AC adapters

€110 / £95 / \$120 approx.

derived from the +5.2 V supply voltage by a TPS7A4901 ultra low-noise linear voltage regulator (IC7). The exact value of the output voltage is set by the voltage divider R28/R29:

$$V_{out} = \left(\frac{R28}{R29} + 1 \right) \times 1.185V$$

For the +3.3 V supply voltage we use the same type of linear voltage regulator (IC9), with the output voltage set by the voltage divider R32/R33.

The negative supply voltage is provided by an LM27761 low-noise inverting voltage regulator (IC8). It uses switched capacitors to invert the input voltage, followed by a low-noise linear regulator which generates a stable output voltage from the inverted input voltage. This output voltage is set to -4.81 V by the voltage divider R30/R31:

$$V_{out} = - \left(\frac{R30}{R31} + 1 \right) \times 1.22V$$

We intentionally chose these "odd" voltages for the output portion (instead of the more common ± 4.5 V) to ensure an undistorted maximum output signal level of 1 V (0 dB).

The power connector (K7) is protected against reverse polarity by the Schottky diode D1. The voltage drop over this diode is just 0.3 V. The current

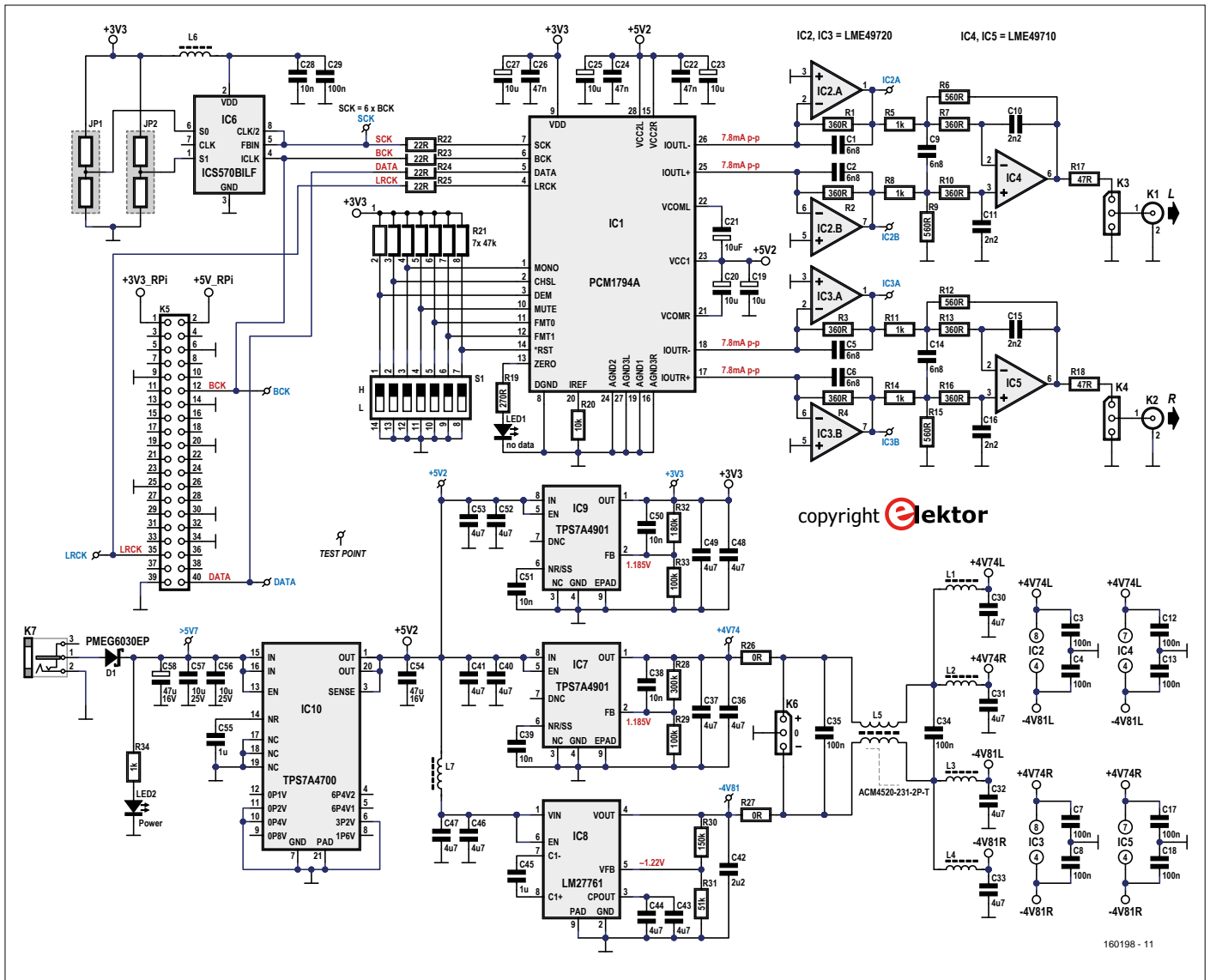


Figure 2. Detailed schematic diagram of the high-end DAC for the Raspberry Pi.

consumption of the audio DAC is listed in **Table 2**.

Master clock

In order to utilize the audio DAC, the operating system of the Raspberry Pi (Raspbian) has to be configured for a HiFiBerry DAC. Once that is done, the Inter-IC Sound (I²S) signals are available on the GPIO expansion connector. The official names of these signals are Continuous Serial Clock (SCK) or Bit Clock (BCLK/BCK), Word Select (WS) or Left/Right Clock (LRCLK/LRCK), and Serial Data (SD/SDATA).

For proper synchronization we also need a master clock, but that is not included in the original I²S specification. This signal is therefore not available on the expansion connector, so the DAC circuit has to generate it on its own. For this

we added a frequency multiplier with an integrated PLL in the form of an ICS570BILF (IC6). This IC has a zero delay buffer to ensure that the rising edges of the input and output signals are perfectly aligned.

The multiplication factor can be set between 0.5 and 32 by two tri-state inputs (S0/S1). According to the data sheet, the output frequency range is 10 to 170 MHz. We chose a multiplication factor of 6 by leaving both S0 and S1 open, and we use the CLK/2 output, which is also connected to the feedback input. This configuration provides the best input range (2.5 to 12.5 MHz) with an output range of 15 to 75 MHz. You might think that with this arrangement it is not possible to handle signals with a sampling frequency of 32 kHz, but that is not true; it works perfectly.

A problem

We ran into a different problem when testing our prototype DAC. First we tried all sorts of 32-bit and 24-bit audio data with sampling frequencies from 32 to 192 kHz, all without any difficulties. However, when we tested the circuit with 16-bit audio data we did not see any output signal. To make a long story short, after a lot of sleuthing we discovered that although the PCM1794A supports 16-bit and 24-bit audio, that is not entirely true in I²S mode. The data sheet is not especially clear about this. Does this mean that our player does not support 16-bit audio? That depends. Direct playback is not possible, but the Volumio version we use here (version 1.55) includes sampling frequency conversion capability with three different quality levels. With a bit of luck, this

capability will also be implemented in the planned update to Volumio 2.

The D/A converter

The DAC device (IC1) has an integrated 8x digital oversampling filter and can handle sampling frequencies from 10 kHz to 200 kHz. The I²S signals are connected through four 22 Ω resistors (R22 to R25) to block RF noise, since

the master clock runs at nearly 74 MHz with a sampling frequency of 192 kHz. The various hardware select lines (MONO, CHSL, DEM, MUTE, FMT0, FMT1 and RESET) are routed to a 7-pole DIP switch and a set of seven 47-kΩ pull-up resistors. This makes changing the settings very easy. The ZERO output on pin 13 drives LED1, which indicates that no audio data is present.

For supply voltage decoupling we use aluminum-polymer capacitors (C19, C23, C25, C27) due to their low equivalent series resistance (ESR) of 40 mΩ at 100 kHz. However, they have the disadvantage of a relatively high leakage current of 100 μA (max.). Since it is not clear whether this leakage current affects the internal input current, we added a pair of normal electrolytic capacitors



COMPONENT LIST

Resistors

- Default: 1%, 0.125W, 0805
 R1,R2,R3,R4,R7,R10,R13,R16 = 360Ω
 R5,R8,R11,R14,R34 = 1kΩ
 R6,R9,R12,R15 = 560Ω
 R17,R18 = 47kΩ
 R19 = 270Ω
 R20 = 10kΩ
 R21 = 47kΩ SIP 7-resistor array, 125mW, 2%
 R22,R23,R24,R25 = 22Ω
 R26,R27 = 0Ω
 R28 = 300kΩ
 R29,R33 = 100kΩ
 R30 = 150kΩ
 R31 = 51kΩ
 R32 = 180kΩ

Capacitors

- C1,C2,C5,C6,C9,C14 = 6.8nF, 63V, 1%, polystyrene, EXFS/HR 6800PF ±1%, LCR Components (alternative 5mm pitch or 0805)
 C3,C4,C7,C8,C12,C13,C17,C18,C29,C34,C35 = 100nF, 50V, 10%, X7R, SMD 0805
 C10,C11,C15,C16 = 2.2nF, 63V, 1%, polystyrene, EXFS/HR 2200PF ±1%, LCR Components (alternative 5mm pitch or 0805)
 C19,C23,C25,C27 = 10μF, 35V, 0.04Ω, diam. 6.3mm max., pitch 2/2.5mm, 870055673001 (WCAP-PTHR series), Würth Elektronik
 C20,C21 = 10μF, 63V, 1.06Ω, diam. 6.3mm max., pitch 2/2.5mm, UPM1J100MDD, Nichicon
 C22,C24,C26 = 47nF, 50V, 10%, X7R, SMD 1206
 C28 = 10nF, 50V, 10%, X7R, SMD 0603
 C30,C31,C32,C33,C36,C37,C40,C41,C43,C44,C46,C47,C48,C49,C52,C53 = 4.7μF, 25V, 10%, X5R, SMD 0805
 C38,C39,C50,C51 = 10nF, 50V, 10%, X7R, SMD 0805
 C42 = 2.2μF, 10V, 10%, X7R, SMD 0805
 C45,C55 = 1μF, 16V, 10%, X7R, SMD 0805
 C54 = 47μF, 16V, 20%, X5R, SMD 1210
 C56,C57 = 10μF, 25V, 10%, X5R, SMD 1206
 C58 = 47μF, 16V, 10%, tantalum, SMD-C (2312), TR3C476K016C0350, Vishay

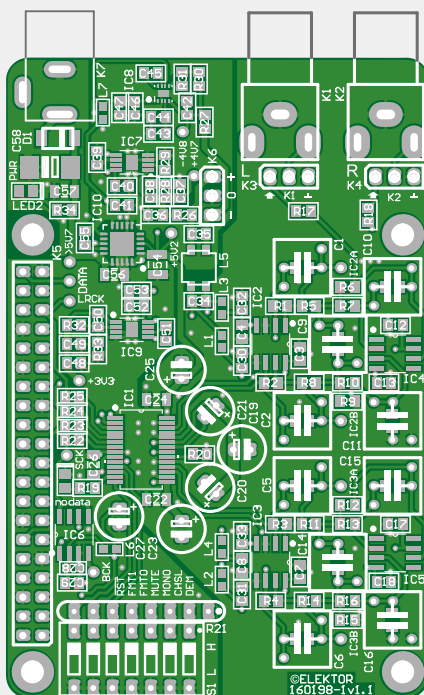


Figure 3. The component side of the PCB.

Inductors and Transformers

- L1,L2,L3,L4,L6,L7 = 600Ω at 100MHz, 0.15Ω, 1.3A, SMD 0603, BLM18KG601SN1D, Murata
 L5 = 2x0.05Ω, 230Ω at 100MHz, 2.6A, SMD, ACM4520-231-2P-T, TDK

Semiconductors

- D1 = PMEG6030EP, 60V, 3A, SMD SOD-128
 LED1,LED2 = green, low power, SMD 0805
 IC1 = PCM1794ADB, SMD SSOP-28
 IC2,IC3 = LME49720MA/NOPB, SMD SOIC-8
 IC4,IC5 = LME49710MA, SMD SOIC-8
 IC6 = ICS570BILF, SMD, SOIC-8
 IC7,IC9 = TPS7A4901DGNT, SMD MSOP-8
 IC8 = LM27761DSGT, SMD WSON-8
 IC10 = TPS7A4700RGWT, SMD VQFN-20

Miscellaneous

- K1 = RCA audio connector, white, PCB mount, right angle, gold plated, PJRANIX1U02AUX, Switchcraft

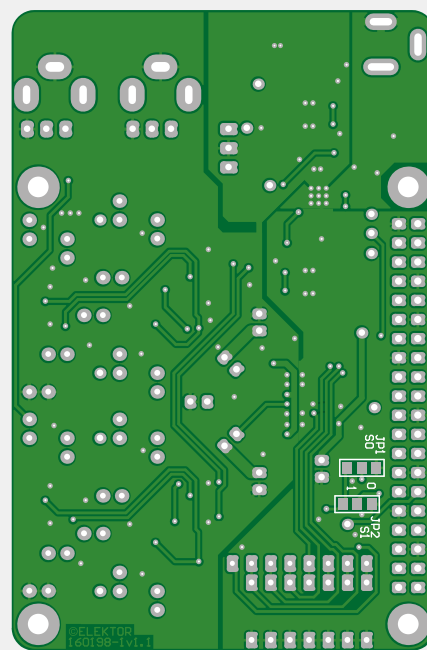


Figure 4. The bottom side of the PCB. Jumpers JP1 and JP2 can normally be omitted.

- K2 = RCA audio connector, red, PCB mount, right angle, gold plated, PJRANIX1U03AUX, Switchcraft
 K3,K4,K6 = 3-pin pinheader, vertical, 0.1" pitch
 K3,K4 = jumper, 0.1" pitch
 K5 = 40-pin GPIO stacking header, 2x20-way female, extra tall
 K7 = DC power connector (jack), 3A, 1.95mm, Lumberg NEB 21 R
 S1 = 7-position DIP switch
 4x M2.5 17mm male/female threaded standoff
 4x M2.5 14mm male/female threaded standoff
 8x M2.5 nut
 PCB 160198-1 v1.1

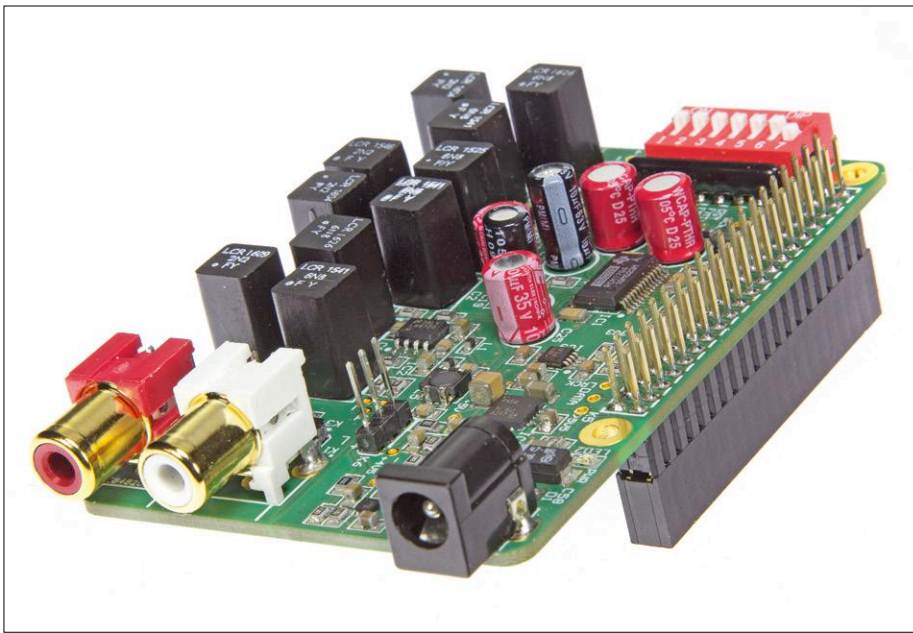


Figure 5. The GPIO connector is mounted on the bottom side.

Table 1: S1 settings		
S1-1	L	De-emphasis disabled for 44.1 kHz
S1-2	L	Digital filter with steep falling edge
S1-3	L	Mono off
S1-4	L	Mono off
S1-5, S1-6	L	I ² S mode
S1-7	H	Reset off

Table 2: Current consumption at different sampling frequencies	
Sampling frequency	Current consumption (K7)
No data	120 mA
32 kHz	127.5 mA
44.1 kHz	131.7 mA
48 kHz	133 mA
96 kHz	149.5 mA
192 kHz	182.1 mA
(Measured with 8 V lab power supply; both outputs terminated with 10 kΩ)	

Table 3: Distortion				
THD+N				
Sampling frequency		48 kHz	96 kHz	192 kHz
1 kHz	BW = 22 kHz	0.0008%	0.0009%	0.0013%
	BW = 80 kHz	0.0028%	0.0012%	0.0014%
7 kHz	BW = 22 kHz	0.00095%	0.0011%	0.0013%
	BW = 80 kHz	0.003%	0.0014%	0.0016%
IMD				
50 Hz:7 kHz = 4:1		0.0014%	0.002%	0.0036%

(ESR 1.06 Ω at 100 kHz, leakage current 4 μA). If you want to experiment with different capacitor types, there is plenty of room on the PCB.

Output filter

Although the sampling artifacts are largely suppressed by the 8x digital oversampling filter, there are still undesirable high frequency components present in the output signal of the DAC. We use a reconstruction filter to suppress these components. The filter in question is a third-order Butterworth filter, and the current to voltage converters (IC2/IC3) are part of this filter. The corner frequency is 64.5 kHz. With a sampling frequency of 44.1 kHz, the 8x frequency component at 352.8 kHz is attenuated by more than 40 dB, and with a sampling frequency of 192 kHz the attenuation is more than 80 dB. The chosen corner frequency is a compromise between adequate attenuation at low sampling frequencies and the audio bandwidth of the output signal at higher sampling frequencies. Here the filter bandwidth is more than three times the normal audible range.

The DAC IC has differential current outputs, which allows an external filter to be used. To convert the output current into a single-ended voltage, we need a current to voltage converter, which in this case is built around the dual opamps of IC2 and IC3. The differential output voltages of IC2A/IC2B or IC3A/IC3B are fed to IC4 or IC5, respectively, which convert them into single-ended voltages. They are also configured as second-order filter stages. In combination with the current to voltage converters, the overall filter characteristic is equivalent to a third-order Butterworth filter.

On the PCB the single-ended output signals are available on connectors K1 and K2 via headers K3 and K4. Jumpers allow the signals from K3 and K4 to be fed directly to K1 and K2. It is also possible to connect a stereo potentiometer to the headers to provide a simple volume control.

The DAC IC does not have integrated volume control capability. Although it is possible to use Volumio for software volume control, that comes at the expense of the resolution and therefore affects the sound quality. For us that is out of the question. We therefore decided to add hardware volume control in the form of a separate board, which also

allows the volume to be adjusted under remote control. The volume control board will be described in the next issue of *Elektor*.

As a final detail, the balanced analog supply voltages for the filter and current to voltage stages are decoupled from the digital supply voltage by a common-mode choke and four RF chokes (for each channel individually).

The board

For the DAC board we only need a single external supply voltage in the range of 7 to 8 V, since the Raspberry Pi board has its own 5 V supply. An external supply voltage of 9 to 12 V is possible, but not recommended due to the higher power dissipation in IC10.

The top and bottom PCB layouts for the DAC board are shown in **Figure 3** and **Figure 4**. The board has the same dimensions as the Raspberry Pi. The power connector and the two signal outputs are on the same end of the board, with the seven-pole DIP switch S1 on the opposite end (adjacent to the WiFi antenna on the RPi 3 board). The 40-pin GPIO stacking header K5 is mounted on the bottom of the board, as can be seen from the photos of the assembled board. The pins of connectors K1–K4, K6 and K7 protruding on the bottom of the board must be trimmed as short as possible to avoid shorting on the metal shells of the Ethernet and USB connectors on the Raspberry Pi board. It is probably a good idea to place a bit of insulation between the two boards. In our prototype we used M2.5 metal male/female threaded standoffs with a length of 17 mm (0.675") to attach the DAC board to the Raspberry Pi. The solder pads for the capacitors in the filter stages are laid out to allow three different types of capacitor to be mounted for C1, C2, C5, C6, C9–C11, and C14–C16. SMD 0805 components can be used here, but it is also possible to mount conventional through-hole capacitors with a lead pitch of 5 mm. For best results, we recommend using 1% polystyrene capacitors.

Pin 1 of each IC is marked with a small white dot, but with IC7 this is difficult to see due to capacitor C38. There was not enough room to put the numbers of the 0805 resistors and the 0805 and 1206 capacitors next to the components concerned, so we put them inside the component outlines. That means they

are hidden on the board. If you assemble the board manually using a hot-air soldering iron or a reflow oven, it's a good idea to use an enlarged copy of the component layout to help you put all the components in the right place. With many of the 0805 components, replacement is difficult or virtually impossible after the conventional through-hole components

have been soldered in place. The latter should therefore be kept for last. Jumpers JP1 and JP2 are located on the bottom of the PCB. They can normally be left open ($SCK = 6 \times BCK$). The audio output connectors (Cinch/RCA connectors) are Switchcraft components with two terminals closer together than in the standard 10 mm version. This means

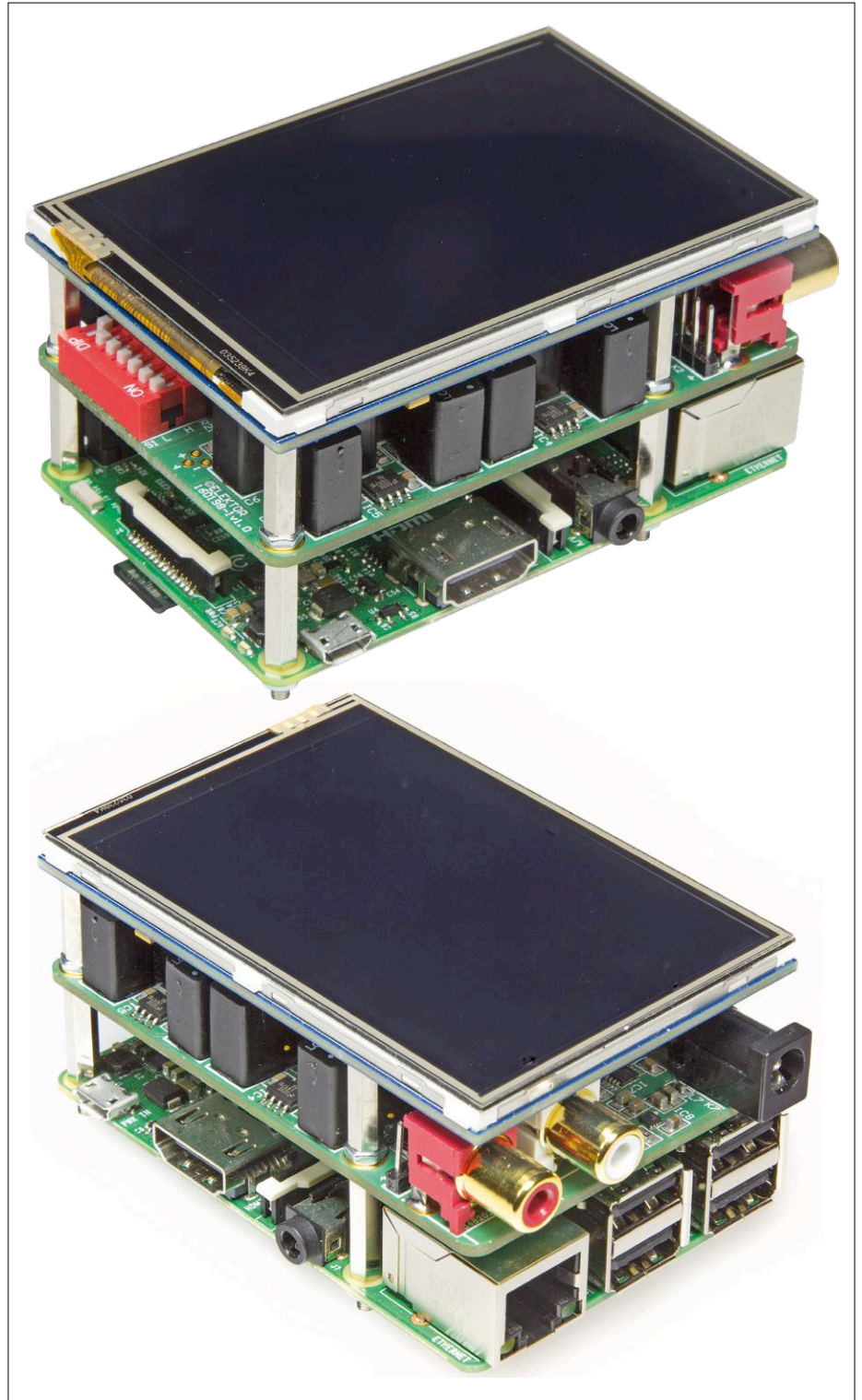


Figure 6: The complete network audio player.

Volumio

Volumio is a free open source Linux distribution specifically tailored to music playback. Volumio runs on a wide variety of devices, including small inexpensive computers such as the Raspberry Pi. After you install Volumio, the device turns onto a headless audiophile music player. Here “headless” means that you have to use another device, such as a smartphone, computer or tablet, to operate the audio player.

For that you can use the Volumio user interface, which is a web app that runs on any device with a browser and makes playing music files easy and intuitive. The web app and Volumio communicate with each other over the local area network. On the Elektor Labs page for this project [2] we provide detailed instructions for installing Volumio on your Raspberry Pi (please note that we used version 1.55) so that you can **also** use the touch screen for this purpose. ×

Figure 7 shows the Volumio user interface on the display (left) and on a PC monitor (right).

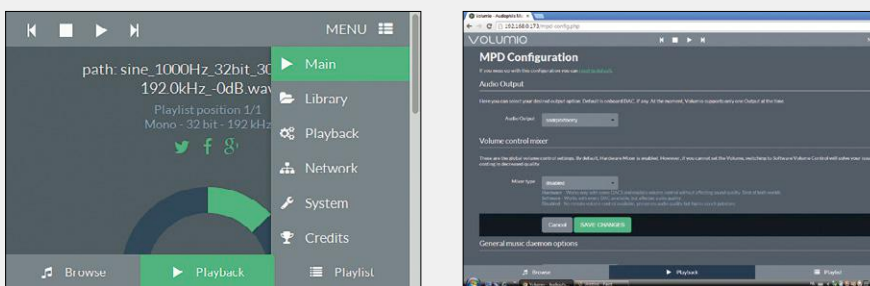
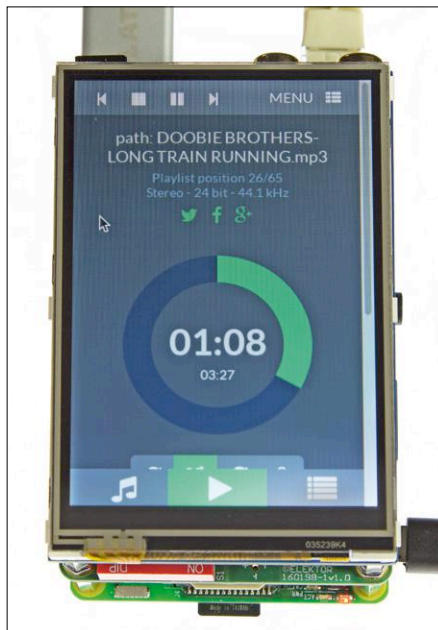
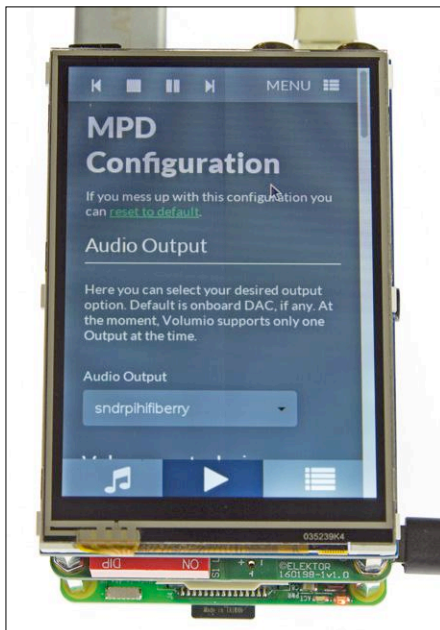


Figure 7. Volumio can be operated using the touchscreen (left) or a PC (right).



FROM THE STORE

 → **160198-1:**
Bare PCB

→ **160198-91:**
Assembled module with touchscreen;
headers included separately (to be
soldered by user))

→ **SKU 17631:**
Raspberry Pi 3 (model B)

that if you want to use a different type, it probably will not fit.

The Waveshare 3.5 inch touchscreen display for the Raspberry Pi must be mounted 16 mm (0.625”) above the DAC board. For this we used four metal male/female threaded standoffs with a length of 14 mm (0.55”) together with an M2.5 nut under each standoff for the extra distance. This means you should not press the connector down as far as possible. In case of doubt, it is better to use a second stacking header and mount the display a bit higher. The distance between the 26-pin connector of the display and the DAC board is about 2 mm.

Figures 5 and 6 show what the complete network audio player looks like.

Measurements

It goes without saying that we subjected our audio DAC to extensive testing. The key results are summarized in **Tables 2 and 3**. All measurements were made with the S1 settings listed in **Table 1**. The current consumption increases at higher sampling frequencies, as can be seen from **Table 2**.

The measured figures for total harmonic distortion plus noise (THD+N) and intermodulation distortion (IMD) are listed in **Table 3**.

Several plots made with our Audio Precision analyzer are also available on the Elektor Labs page for this project [2]. ◀

(160198-1)

Web links

- [1] PCM1794A data sheet: www.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=pcm1794a&fileType=pdf
- [2] Elektor Labs project page: www.elektormagazine.com/labs/audio-dac-for-rpi-networked-audio-player-using-volumio

- More than 45 years of experience
- 24-hour shipping
- More than 75,000 products

INDEPENDENT POWER SUPPLY IDEAL FOR TRAVELLING

(Intenso)[®]

Intenso slim 10,000 mAh power bank

Our price tip – ideal for smartphones & tablets!

- Cell capacity: 10,000 mAh, lithium-ion
- Output connection: USB A and integrated micro USB cable
- Output: 5 V DC, max. 2.1 A
- Protective features: surge voltage protection, discharge protection, overcharge protection, short-circuit protection
- Dimensions: 130 x 70 x 16 mm

Order no.: INTENSO 7332530
instead of 16.40

SAVE 41% **9.68**



TO HELP YOU CHOOSE

 <p>Smartphone suitable for power banks with 1–1.5 A and 2,200–6,000 mAh</p>	 <p>Tablet PCs suitable for power banks with more than 1.5 A and 6,000–20,000 mAh</p>
--	---

Stabilised universal adapter, 12–24 V DC

Supplies netbooks and notebooks with reliable power while travelling by car!

gobay[®]

- Input voltage: 12–24 V DC
- Output: 15–20 V DC / max. 4,730 mA
- 10 DC connector plugs included for different devices
- Short-circuit, overload, surge current and overheating protection
- Integrated USB charging port, 5 V DC



Order no.: NTS 90 USB

25.86

Logilink slim 20,000 mAh power bank

The power bank with huge power reserves

- Output: 5 V DC, max. 2.1 A
- LED charge state display (4 segments)
- Protective functions: overcharge protection, deep discharge protection, short-circuit protection
- 2 USB ports
- Dimensions (L x W x H): 160 x 82 x 23 mm

HOW ABOUT SOMETHING EXTRA?!

Order no.: LOGILINK PA0086B

22.38






OUR HOTLINE NOW ALSO IN YOUR LANGUAGE

We look forward to your call:
+44 203 808 95 25



Daily prices! Price as of: 29. 5. 2017

Prices in £ plus statutory VAT, plus shipping costs · reichelt elektronik, Elektronikring 1, 26452 Sande (Germany)

Onlineshop languages:   

PAYMENT METHODS:



SHOP CONVENIENTLY ONLINE!



www.reichelt.co.uk

 ORDER HOTLINE: +44 203 808 95 25

BBC micro:bit for Electronicists (2)

Data acquisition and oscilloscope functions

By **Burkhard Kainka** (Germany)

Every microcontroller that features an A-to-D converter and a PC interface can be used as a logging device for data acquisition systems. But with the BBC micro:bit you get the bonus of a small LED display and a wirefree interface into the bargain. Just the job for special applications in your electronics lab!

The small footprint of the BBC micro:bit would alone make it a go-to choice for measurement tasks. Regardless of whether powered over a USB cable or by batteries, and whether it used Bluetooth or some simplified wireless protocol to communicate with the outside world, it can always be installed close to the object under examination.

A USB oscilloscope

For programming the BBC micro:bit the mbed platform has proven its worth. The

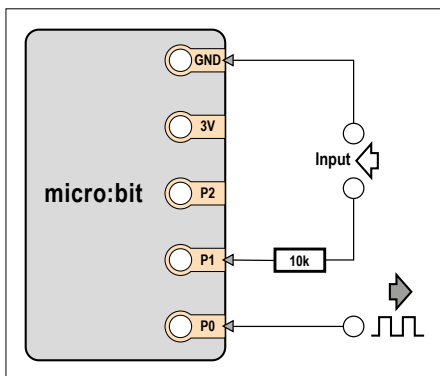
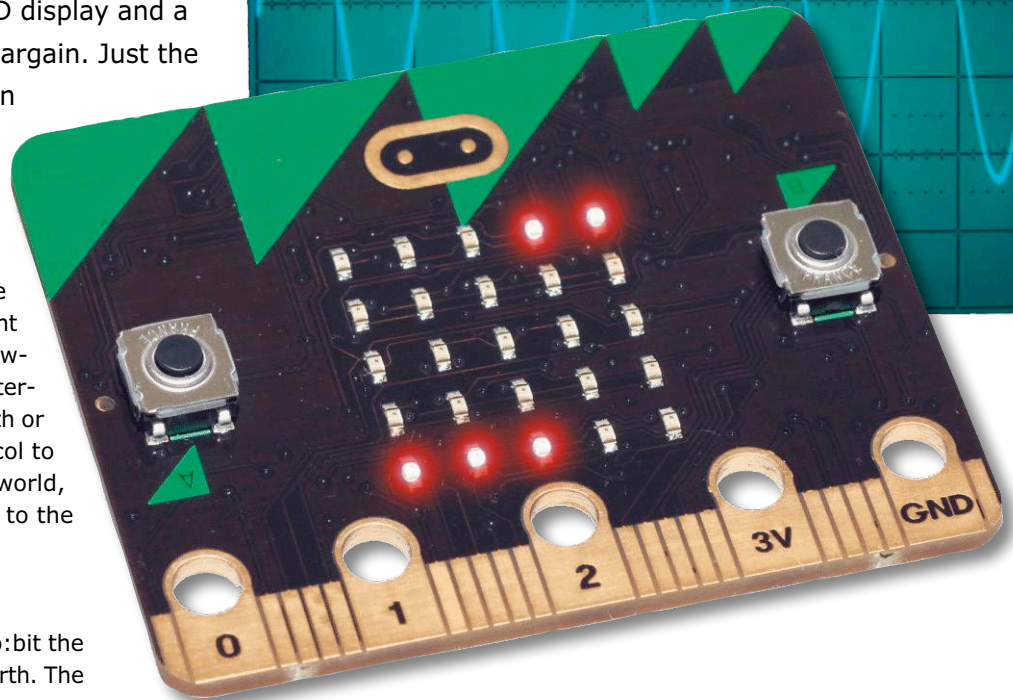
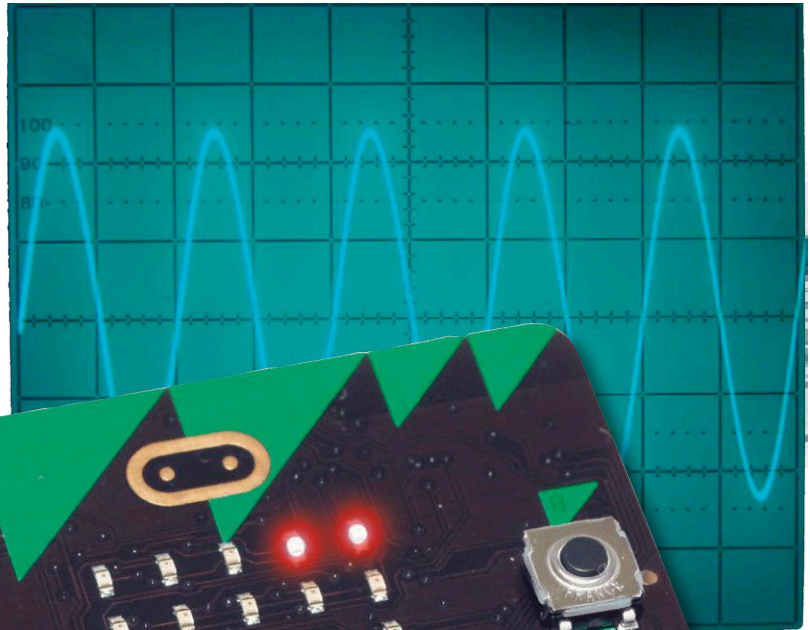


Figure 1. Measurement input and signal output.

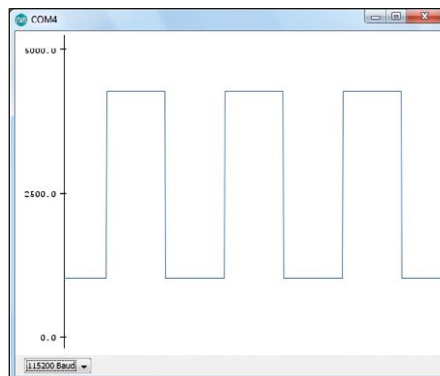


Figure 2. A squarewave signal with 10 Hz repetition rate.

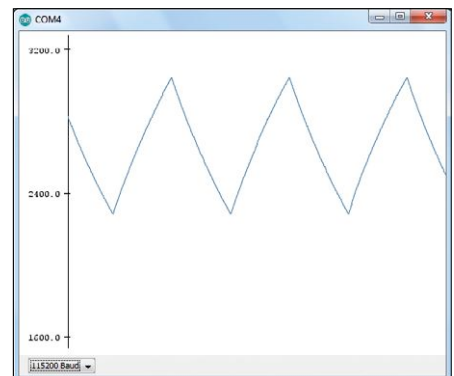


Figure 3. The filtered signal.

basics are explained in reference [1]. All the programs mentioned in this article are available to download as text files on the Elektor website [2] and need to be copied into an existing mbed project. When deploying the BBC micro:bit for general-purpose measurements you must ensure that the voltage range under examination cannot overstep the range between GND and V_{cc} . A protective resistor of 10 k Ω in series with the input will restrict the current flow in all situations, necessary if you accidentally exceed safe limits (**Figure 1**). As well as the analog input there's also a square-wave output, with which you can produce a handy test signal.

For maximum speed the program in **Listing 1** captures data without buffering for immediate transfer of each measured value. A significant element of the cycle time arises from the serial transfer at 115,200 baud. If the measurements captured are a mixture of, say, single-digit (3 mV) and four-digit (3000 mV) amounts, these variable figures will result in uneven data flow. Consequently we raise the voltage by 1000 mV, making the possible values from 1000 mV to 4300 mV and always taking the same time to process. The program also provides its own signal source so that you can measure something without additional overheads. P1 becomes a PWM output with a PWM frequency of 10 Hz and a period of 100 ms. For evaluating the data there are plenty of options. You could take in the data using a terminal program and then present it as a spreadsheet. A convenient alternative is the serial plotter in the Arduino IDE (version 1.6.8 onwards). This software provides a scrolling screen display, adjusted automatically to the display range, making range switching or reformatting unnecessary. With the settings shown we can measure a symmetrical squarewave signal with 10-Hz repetition rate (**Figure 2**). At the same time we now have a known time axis. The entire oscillogram clearly depicts a data acquisition duration of 300 ms. The serial plotter of the Arduino IDE always plots first from left to right until the screen is completely filled. After this, the picture scrolls to the left to make old data disappear. The choice between a continuous or static display is made with the button A (`if(uBit.buttonA.isPressed()` in Listing 1). Just press button A for the duration of the measure-

Listing 1. Rapid measurement with direct data transfer.

```
//Voltage Logger/Scope
#include "MicroBit.h"
MicroBit uBit;

int main()
{
    uBit.init();
    MicroBitSerial serial(USBTX, USBRX);
    uBit.io.P1.setAnalogValue(512);
    uBit.io.P1.setAnalogPeriodUs(100000);
    while (1) {
        if(uBit.buttonA.isPressed()){
            int u = 1000+3300 * uBit.io.P0.getAnalogValue()/ 1023;
            uBit.serial.printf("%d\r\n", u);
            // uBit.sleep(100);
        }
    }
}
```

ment and a scrolling display is shown. As soon as you release the button, the last display is frozen, so you can examine it more closely or save a copy of it. The additional signal output can be useful for investigating circuits or components. Using a low-pass filter with 4.7 k Ω and 22 μ F a sawtooth signal is gener-

ated from the squarewave, as would be expected (**Figure 3**). The measurement range of the Arduino plotter adapts automatically to smaller voltages.

Faster sampling by buffering

If you are minded to reduce the time taken by serial data, it is only the acqui-

Listing 2. Rapid saving and subsequent transfer.

```
//Fast Scope
#include "MicroBit.h"
MicroBit uBit;

int main(){
    char d[400];
    uBit.init();
    MicroBitSerial serial(USBTX, USBRX);
    uBit.io.P1.setAnalogValue(512);
    uBit.io.P1.setAnalogPeriodUs(2000);
    while (1) {
        if(uBit.buttonA.isPressed()){
            for(int i = 0; i < 400; i++){
                d[i] = uBit.io.P0.getAnalogValue()/ 4;
            }
            for(int i = 0; i < 50; i++) uBit.serial.printf("%d\r\n", 0);
            for(int i = 0; i < 400; i++){
                uBit.serial.printf("%d\r\n", d[i]);
            }
            for(int i = 0; i < 50; i++) uBit.serial.printf("%d\r\n", 255);
        }
        uBit.sleep(500);
    }
}
```

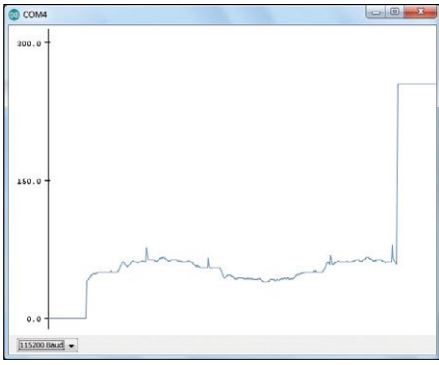


Figure 4. Measuring with a higher sampling rate.

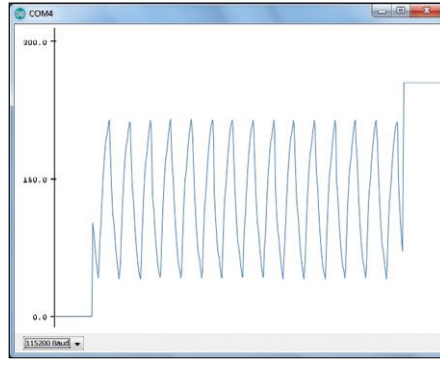


Figure 5. A 500-Hz sawtooth signal

sition time of the A-to-D converter that limits the achievable sampling rate. So you create a Data Array, fill it with data measurements and send these off to the PC. But it's exactly here that problems arise that you had not reckoned with. Although the controller is well endowed with RAM, an Array of the type `int d[100]` is pushing the limits, because the Microbit Runtime does not have much spare capacity. But if you want to use the serial plotter, there should already be 500 measured values.

You definitely need to be aware that the type of `int` in a 32-bit system has a size of four bytes. Accordingly we have 400 bytes at our disposal. Therefore we

use a Byte Array with 400 bytes using `char d[400]`. By dividing by 4, the 10-bit data of the A-to-D converter becomes an economical 8 bits.

We now store and transfer 400 bytes (**Listing 2**). We are still 100 bytes short for filling the plotter. But we can make a virtue out of necessity and prefix a zero-bytes header and suffix a trailer having 255-bytes. This causes the serial plotter to always display the full measurement range and provides the viewer with clear visibility of the range limits. The reading in **Figure 4** depicts a 50-Hz signal with typical interference pulses, which you pick up with an exposed measurement lead. Because around 1.5 oscillations are

shown, the measurement lasts around 30 ms, which means that with 400 data points a sampling rate of approximately 13 kHz can be deduced. Because the PWM frequency in this program was raised by 500 Hz, you can easily verify this with a signal of your own. **Figure 5** shows the PWM signal at the output a low-pass filter with 4.7 kΩ and 100 nF.

Wireless transfer of captured data

The BBC micro:bit is equipped with Bluetooth Low Energy (BLE). You will look in vain on the board for a dedicated chip for this, as the RF circuitry is already built into the microcontroller. The nRF51822 from Nordic Semiconductor was originally developed for applications such as wireless keyboards and mice, which did not call for extensive range but did have to use battery power economically. The BBC micro:bit takes advantage of these capabilities. You can power the board using a 3-V battery and then dispense with even the USB cable. This makes the system viable even for long-term applications powered by batteries. The shortform lineup of its main features speaks for itself:

- 2.4-GHz transceiver
- -93 dBm sensitivity in Bluetooth® low energy mode
- 250 kbps, 1 Mbps, 2 Mbps supported data rates
- Tx Power -20 to +4 dBm in 4-dB steps
- Tx Power -30 dBm whisper mode
- 13 mA peak Rx, 10.5 mA peak Tx (0 dBm)
- 9.7 mA peak Rx, 8 mA peak Tx (0 dBm) with DC/DC
- RSSI (1-dB resolution)
- ARM® Cortex™-M0 32-bit processor
- 275 μA/MHz running from flash memory
- 150 μA/MHz running from RAM
- Serial Wire Debug (SWD)

Programming with mbed enables the use of Bluetooth, allowing you to transmit data direct to a smartphone or tablet. Admittedly this calls for a pretty sizeable software stack and worse, it leaves you to develop the custom apps.

But there's a far simpler way. You see, you can address the 2.4-GHz transceiver at a lower level, dispensing altogether with the complicated Bluetooth protocol. Making this possible is the MicroBitRadio

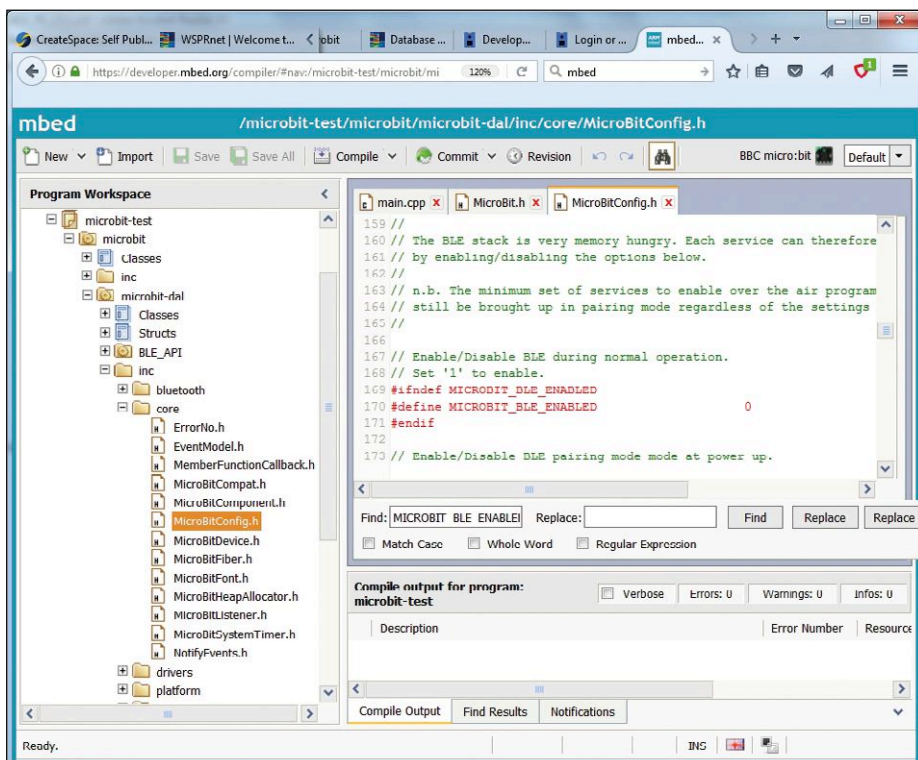


Figure 6. Deactivating BLE.

support platform [6] with simple datagrams (text messages) that the transceiver can handle unaltered. One BBC micro:bit module sends a text message and all other modules in range can receive it. To make this work a default channel and a default transmit power have been defined, so you really don't need to concern yourself with anything. That makes this one of the simplest methods of transferring data without wires. At the same time it gives you the ability to make isolated (potential-free) measurements. A typical application might be an electrocardiogram device, as the electrical (galvanic) separation eliminates any troublesome electrical hum interference. All you need is two BBC micro:bit modules. One is employed as the measuring instrument for transmitting the data, the second one is programmed as the receiver, for displaying the data or transferring it to a PC over a USB cable. To use MicroBitRadio in mbed you do need to deactivate Bluetooth Low Energy. There's a note about this in the BBC micro:bit documentation: *It is not currently possible to run the*

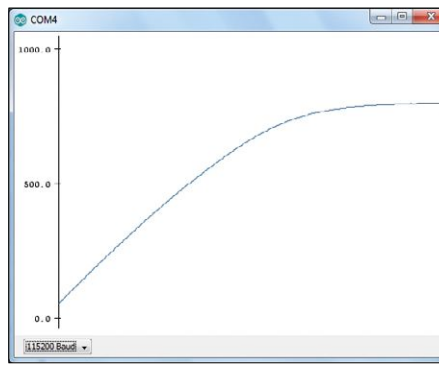


Figure 7. Data measurements sent by wireless link.

MicroBitRadio component and Bluetooth Low Energy (BLE) at the same time. If you want to use the MicroBitRadio functionality, you need to disable the BLE stack on your micro:bit by compiling the runtime with #define MICROBIT_BLE_ENABLED 0 in your inc/MicroBitConfig.h file.

It is not entirely simple to locate the correct point in the numerous files of the runtime system. The exact path is: *microbit\microbit-dal\inc\core\MicroBitConfig.h* (see **Figure 6**). In this

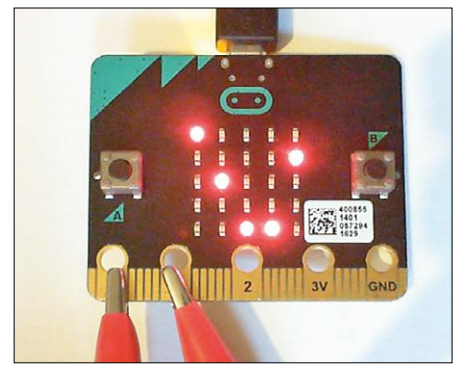


Figure 8. Measuring the internal test signal.

file is an entry `MICROBIT_BLE_ENABLED 1`, in which you need to replace the 1 with a 0. In the next compilation that you make BLE is then deactivated, enabling you to use the simplified MicroBitRadio. The program in **Listing 3** sends and receives datagrams of measurement data. The sender and receiver can therefore use the same program and also exchange data reciprocally. The sender executes a measurement at P1 and transmits the reading in mV. The receiver passes the received data forward via the

Advertisement

Join the Elektor Community

Take out a GOLD Membership now!

Also available:
The all-paperless GREEN Membership!
www.elektor.com/member

GOLD MEMBERSHIP

- ✓ 6x Elektor Magazine (Print)
- ✓ 6x Elektor Magazine (PDF)
- ✓ Access to Elektor Archive (Thousands of Articles)
- ✓ Access to over 1,000 Gerber files
- ✓ Elektor Annual DVD
- ✓ 10% Discount in Elektor Store
- ✓ Exclusive Offers

GREEN MEMBERSHIP

- ✓ 6x Elektor Magazine (PDF)
- ✓ Access to Elektor Archive (Thousands of Articles)
- ✓ Access to over 1,000 Gerber files
- ✓ 10% Discount in Elektor Store
- ✓ Exclusive Offers

USB connection. Nothing changes on the PC side. Here we can again make use of the serial plotter. **Figure 7** illustrates a measurement reading. The transmitter is battery-driven and is located three meters (10 feet) away from the receiver. The link works for up to approx. 10 m (30 ft.). A 10- μ F electrolytic was attached to analog input P1. This was charged up slowly using the 10-M Ω pull-up resistor provided on the PCB. You may notice some deviation from the normal charging curve, because this capacitor had not been used for a long time. In a case like this just a low leakage current flows initially that increases only gradually. On the right-hand side of the diagram the measured voltage lies clearly below 1 V and rises only slowly.

Mini-oscilloscope with LED display

An extremely basic oscilloscope is better than none at all and sometimes it's more important that the device is very small, standalone and easy to handle. Here we see measurement data displayed graphically on the LED display using 5x5 LEDs (**Listing 4**). Even if you are accustomed to a far more sophisticated instrument, you can definitely get results with this little alternative. It is quite remarkable, what is still discernible with such a simple 'scope.

Once again the mini-oscilloscope uses Port 1 as an analog input and additionally employs Port 0 as a PWM output. With a repetition rate of 500 μ s, an output signal with a frequency of 2 kHz is generated. A direct connection to the measurement input shows the limits of the A-to-D converter (**Figure 8**). The sampling time is obviously too long to display sharp edges of the PWM signal. The limiting frequency of this simple oscilloscope is therefore somewhere below 10 kHz. This is not enough for an RF lab but probably adequate for many simple measurements and experiments. ◀

(160384)

Web Links

- [1] www.elektormagazine.com/160273
- [2] www.elektormagazine.com/160384
- [3] <https://developer.mbed.org/>
- [4] <https://lancaster-university.github.io/microbit-docs/ubit>

Listing 3. Sending and receiving datagrams.

```
//Radio Data
#include "MicroBit.h"
MicroBit uBit;

void onData(MicroBitEvent e)
{
    ManagedString s = uBit.radio.datagram.recv();
    uBit.serial.send (s);
    uBit.serial.send (" \r\n");
}

int main()
{
    uBit.init();
    uBit.messageBus.listen(MICROBIT_ID_RADIO, MICROBIT_RADIO_EVT_
DATAGRAM, onData);
    uBit.radio.enable();
    char output[16];
    while (1) {
        int u = 3300 * uBit.io.P1.getAnalogValue()/ 1023;
        itoa (u, output);
        uBit.radio.datagram.send(output);
        uBit.sleep(100);
    }
}
```

- [5] [B. Kainka, BBC micro:bit Tests Tricks Secrets Code, CreateSpace 2016](#)
- [6] <https://lancaster-university.github.io/microbit-docs/ubit/radio/>

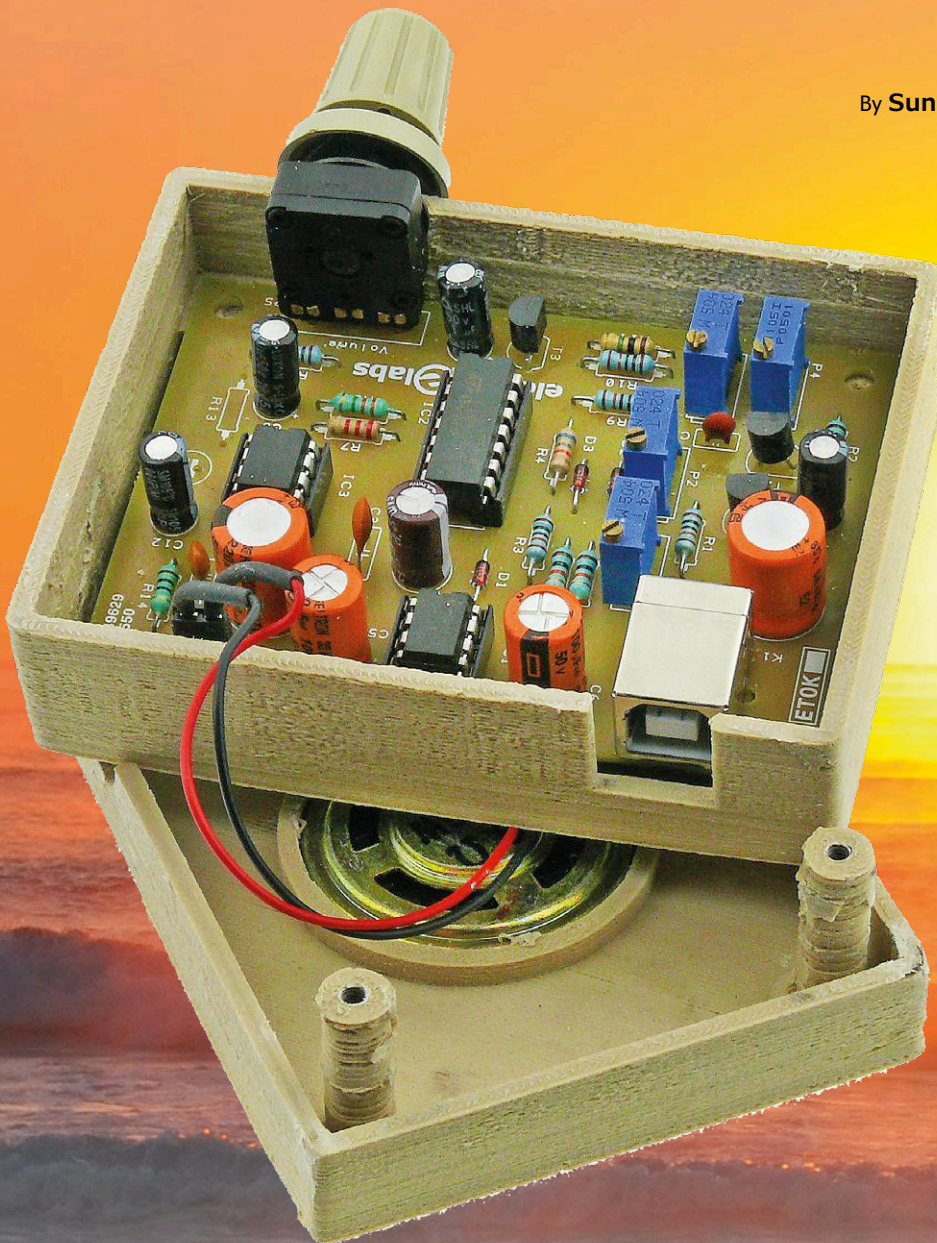
Listing 4. Using the LED display.

```
//LED-Scope
#include "MicroBit.h"
MicroBit uBit;
int main()
{
    int y;
    uBit.init();
    uBit.io.P0.setAnalogValue(512);
    uBit.io.P0.setAnalogPeriodUs(500);
    uBit.display.enable();
    MicroBitImage image(5,5);
    while (1) {
        for(int x = 0; x < 5; x++){
            y = 4- (uBit.io.P1.getAnalogValue()/205);
            image.setPixelValue(x,y,255);
        }
        uBit.display.print(image);
        uBit.sleep(500);
        image.clear();
    }
}
```


Sea Murmur Simulator

Gentle noise to help you go to sleep

By **Sunil Malekar** and **Clemens Valens** (Elektor Labs)



When Morpheus' arms keep eluding you, listening to the sound of surf and waves breaking on a beach may help you to relax. With the circuit presented here switched on at your bedside Mr Sandman is sure to stop by.

You may think that for producing a magazine like *Elektor* all that is required are some editors to write articles and one or two graphics designers to create nice illustrations. That is only partly true. One of the main reasons — possibly the only one — for you to be able to read your favorite magazine every eight

weeks is because of our Multifunctional Power Planner and unsung hero Hedwig. Although electronics is not her specialty ("is this capacitor thing or whatever it is supposed to be this kind of blue?"), Hedwig does have a pet circuit: the sea sound generator. So Hedwig, this one is for you.

PROJECT INFORMATION



Noise
Simulator
Relaxation



entry level
intermediate level
expert level



1 hour approx.



Soldering iron,
small screwdriver,
bed



£12 / €15 / \$17 approx.

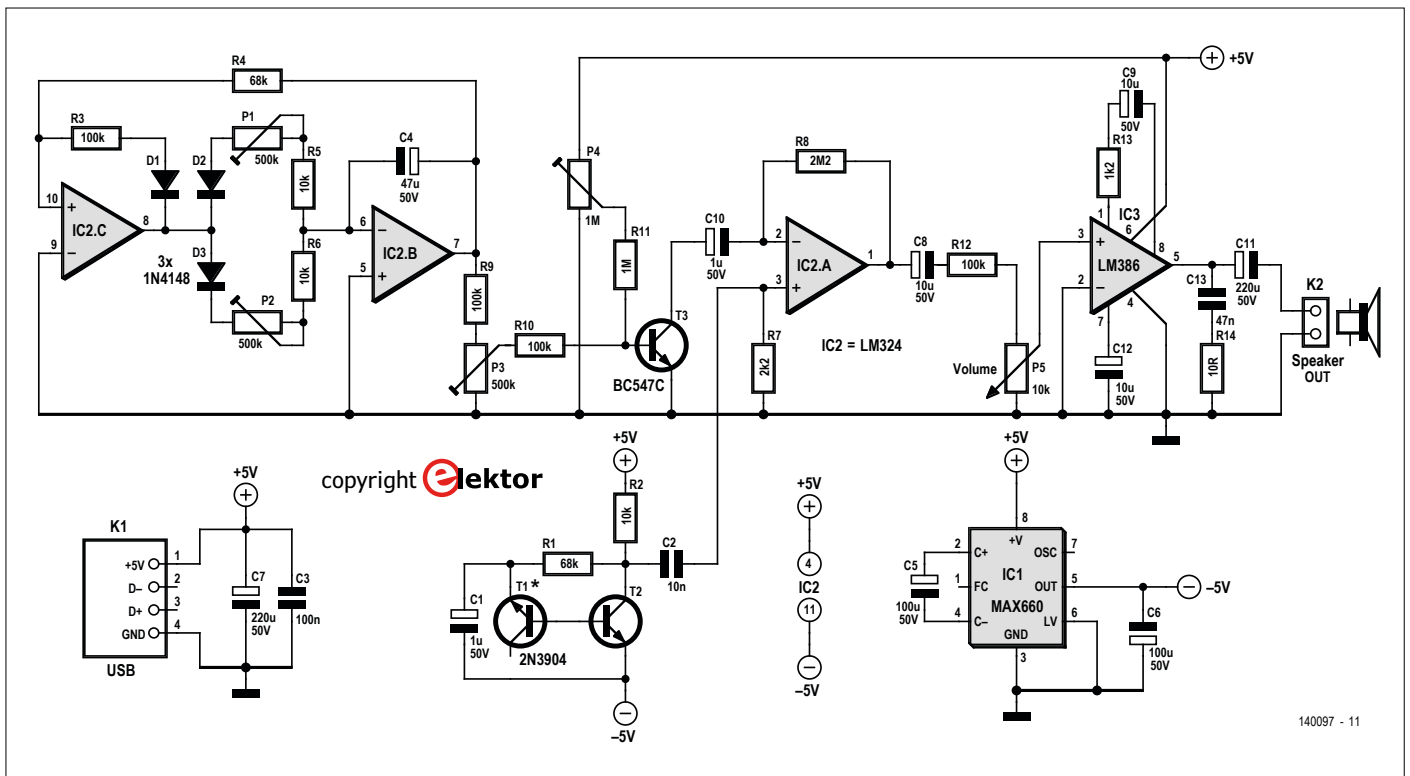
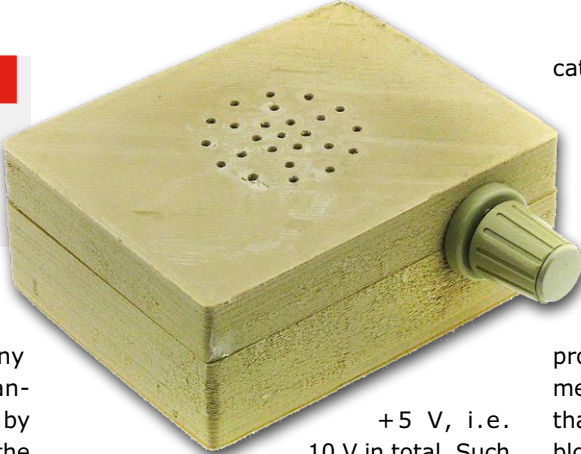


Figure 1. The internals of the sea murmur simulator.

- ### Features
- Independent control of wave attack and decay shape
 - Powered from a phone charger
 - No (0) Arduino inside



The sound of waves breaking on the beach is considered relaxing by many people. As relaxing and even romantic as it may appear, when analyzed by the impartial and objective scientist the sound turns out to be modulated noise in essence. Perhaps slightly pink noise, as a vague reminiscence of the setting sun, but noise it is. Noise is easy to produce with an electronic circuit — as a matter of fact, it is much more difficult to not produce noise with electronics. Basically, all you need to do is reverse-bias a diode.

The circuit

Even though generating noise is easy, making usable noise requires some effort. **Figure 1** shows the schematic of the contraption we came up with. Here the noise source is T1 wired as a reversed-biased diode with its emitter mimicking a cathode and its base, an anode. The noise it produces is buffered by T2. Looking closely you will notice that this transistor is connected between -5 V and

+5 V, i.e. 10 V in total. Such a 'high' voltage is necessary for producing good noise. T1 and T2 are specified as 2N3904 because it was found that these transistors produce more noise than the more Elektor-savvy BC547 a.k.a. TUN in the old days. We tried that one too, and it worked, but you may obtain better results with the given types. If you want to experiment with these transistors, keep in mind that the 2N3904 for which we designed the printed circuit board does not have the same pinout as the BC547.

Now that we have noise, we have to modulate it to simulate the sound of the waves and the surf. Modulation requires an oscillator. Some sort of asymmetric sine-like wave is likely to result in the most realistic effect, but this compli-

cates matters a bit, especially because of the asymmetrical aspect, i.e. the swell or attack of the sound should not have the same duration as its decay. An asymmetric triangle generator is much easier to build, see the circuitry around IC2.C and IC2.B, and the result is more than satisfying. Normally such an oscillator produces a triangular wave that's symmetrical around zero, but we don't want that here as the negative halves would block our modulator T3, resulting in discontinuous breakers. To prevent this happening, D1 was added. Now the output of IC2.B sweeps nicely between 0 V and about 3 V.

Pots P1 and P2 control the slopes of the modulating signal; P1 adjusts the rising slope, P2 the falling slope. Together they determine the frequency of the signal, the speed of the waves so to speak. The smaller their total value, the higher the frequency. Their ratio determines the symmetry of the signal.

Transistor T3 acts as a current-controlled resistance controlling the amplification of IC2.A; the smaller the resistance, the stronger the output signals. Pots P3 and P4 then, provide a means of fine tuning the amplification.

The output of IC2.A is fed through P5, the overall volume control, to a power amplifier built around IC3. This is a classic LM386 circuit capable of driving a loudspeaker or headphones. The gain of the amplifier can be set to anywhere from 20 to 200 by using the right combinations of parts R13 and C9.

Without these parts the gain is 20, with only C9 between pins 1 and 8 (i.e. R13 = 0 Ω) the gain is maximum (200). The values shown in the schematic set the gain to 50 or so.

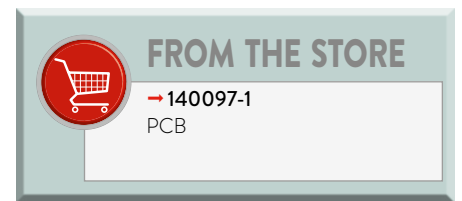
Although a large part of the circuit is powered from a symmetric ± 5 V supply, a standard 5-V USB phone charger is sufficient to make it all work. This is made possible by IC1, a MAX660 chip wired as a switched-capacitor voltage inverter, and USB connector K1.

Calibration

Like every high-precision instrument, our sea murmur simulator has several trim-pots that require adjustment in order to get the best result. Luckily, the procedure is simple. It starts by turning P3 to its minimum position (i.e. wiper to 0 V). P4 controls the background noise of the virtual sea when there are no waves at all. Adjust P4 for a nice, soft whisper in the loudspeaker.

Now adjust P3 to add waves. It effectively controls the amplitude of the waves — simply set it to a realistic level. If the sound appears to start clipping, lessen off on P4 a little. Remember that P3 and P4 affect each other so that further fine tuning may be necessary.

Finally the 'shape' of the waves has to be adjusted with P1 and P2. Extensive



studies and simulations in the lab have shown that the rise of a wave (P1) is usually 5 to 10 times shorter than its decay (P2); the required wave shape is therefore more like a sawtooth than a triangle. The wave period should be several seconds — adjust to taste. ◀

(140097)

Web Link

[1] www.elektormagazine.com/140097



COMPONENT LIST

Resistors

R1,R4 = 68k Ω
 R2,R5,R6 = 10k Ω
 R3,R9,R10,R12 = 100k Ω
 R7 = 2.2k Ω
 R8 = 2.2M Ω
 R11 = 1M Ω
 R13 = 1.2k Ω
 R14 = 10 Ω
 P1,P2,P3 = 500k Ω trimpot
 P4 = 1M Ω trimpot
 P5 = 10k Ω linear potentiometer

Capacitors

C1,C10 = 1 μ F 50V
 C2 = 10nF
 C3 = 100nF
 C4 = 47 μ F 50V
 C5,C6 = 100 μ F 50V
 C7,C11 = 220 μ F 50V
 C8,C9,C12 = 10 μ F 50V
 C13 = 47nF

Semiconductors

D1,D2,D3 = 1N4148
 IC1 = MAX660

IC2 = LM324

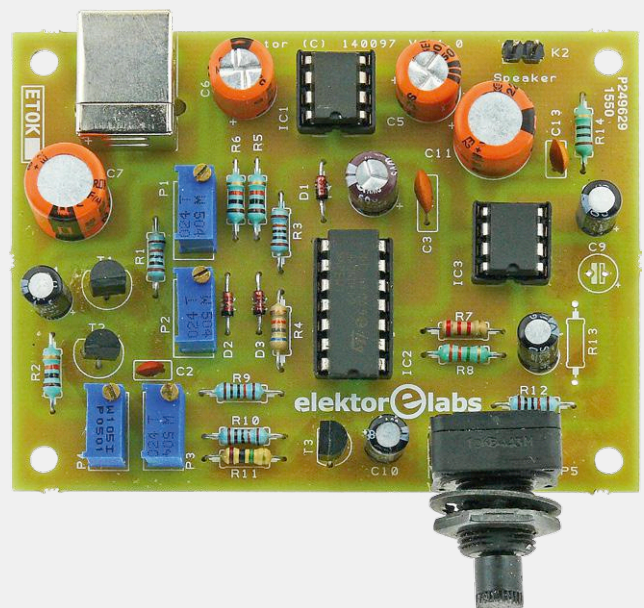
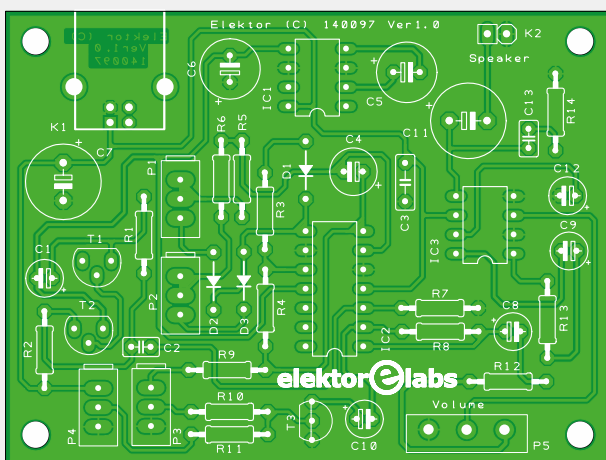
IC3 = LM386

T1,T2 = 2N3904

T3 = BC547C

Miscellaneous

K1 = USB-B connector
 K2 = 2-pin pinheader, 0.1" pitch
 PCB # 140097-1

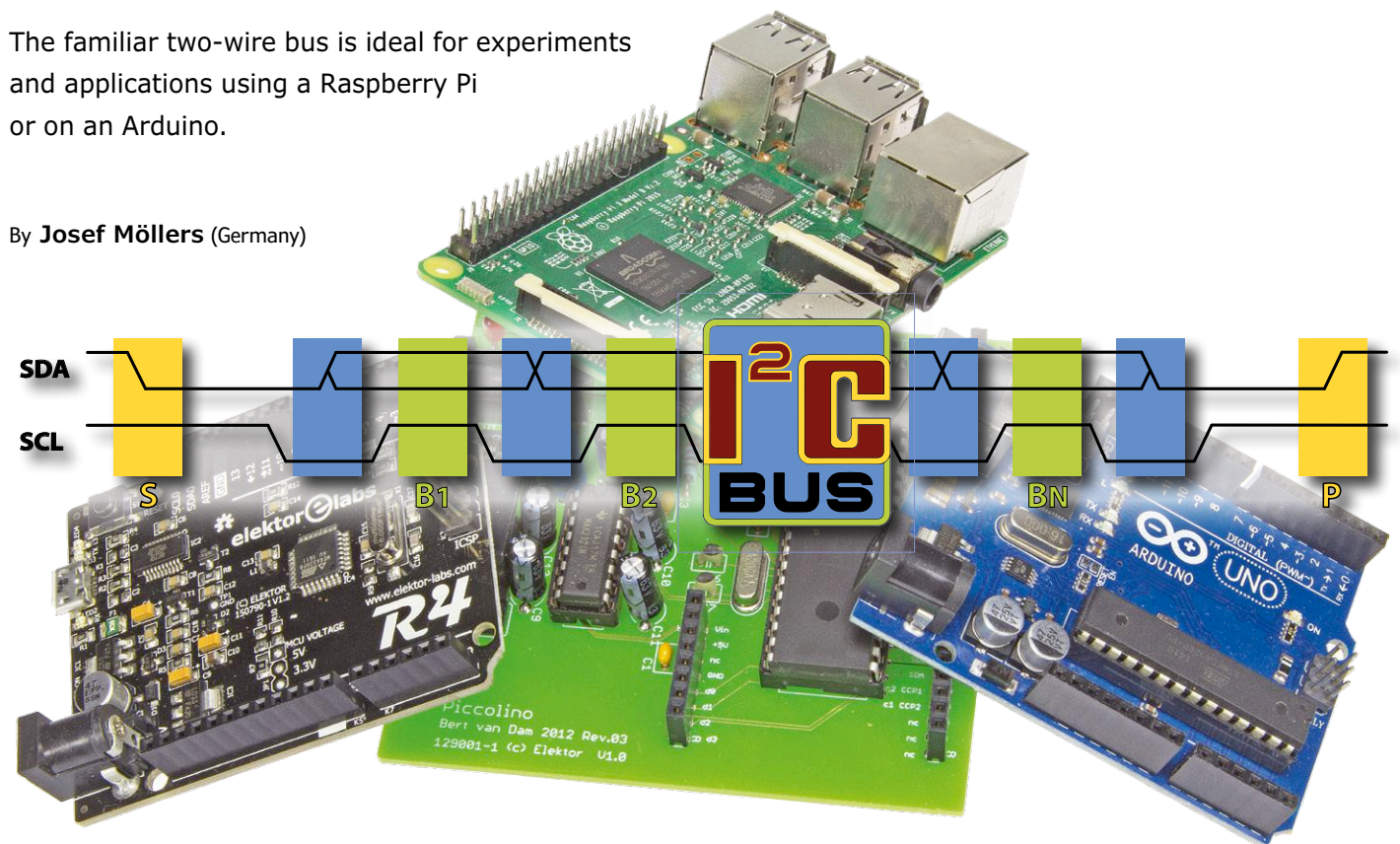


The I²C Bus

Part 2: using the bus with a microcontroller

The familiar two-wire bus is ideal for experiments and applications using a Raspberry Pi or on an Arduino.

By **Josef Möllers** (Germany)



Raspberry Pi, BeagleBone, Arduino, Genuino, ATmega, PIC, practically any PC: pretty much anything you can find sitting on a maker's work bench that can compute will have one or more I²C interfaces. Using the example of the LM75 temperature sensor we will look in this article at how the I²C interfaces of the Raspberry Pi, ATmega and Arduino can be used as bus masters, and, where possible, as bus slaves.

Raspberry Pi

The Raspberry Pi has two physical I²C buses, of which only one can normally be used directly. The Raspberry Pi board comes with pull-up resistors to the 3.3 V supply already fitted and permanently enabled. SDA and SCL are available on pin 3 (SDA) and pin 5 (SCL) of the expansion header, conveniently right next to the 3.3 V supply on pin 1, the 5 V supply on pins 2 and 4, and ground on pin 6. These pins belong to I²C bus number 1. It is therefore possible to make a very compact plug-in expansion board offering temperature sensing, real-time clock or position sensing functions.

Caution: the ports of the Raspberry Pi must **only be operated at 3.3 V** and connection to 5 V signals can damage the device. It is important, therefore, to check your circuit carefully before connecting it to the ports on the Raspberry Pi. In particular take care that no I²C slave contains pull-up resistors to the 5 V rail. If present, any such resistors should be removed: the slave will still work without them.

The Raspberry Pi provides a convenient environment for learning how to use new and unfamiliar I²C slave devices. **Figure 1** shows how a breadboard can be used to connect an LM75 to a Raspberry Pi.

Operation as bus master

Before the I²C bus on the Raspberry Pi can be used under Raspbian, it is necessary to install two extra drivers. To do this, launch `raspi-config` and select the `I2C` option under the `Advanced Options` menu item. This will enable the interface and load the necessary kernel module. Alternatively add the following lines to the file `/etc/modules` using a text editor:

```
i2c-dev
i2c-bcm2708
```

After restarting the system the two drivers (as well as any other extra drivers that are required) will be loaded and the device nodes `/dev/i2c-<n>` will be created. This can be confirmed using the following commands at the '\$...' prompt.

```
$ lsmod | i2c_
i2c_dev          XXXX  0
i2c_bcm2708     YYYYY 0
$ ls /dev/i2c-*
dev/i2c-1
```


In the above XXXX and YYYY stand for the size of the modules, while the two zeros indicate that no programs are currently using the modules. The commands will, incidentally, work even without root privileges.

Next install the `i2c-tools` package, which includes among other things code to detect I²C devices and buses:

```
sudo apt-get install i2c-tools
```

Raspbian already includes drivers from some I²C peripheral devices, including for the RV-8523 real-time clock (RTC). Unless configured otherwise, the Raspberry Pi drives the I²C bus in standard mode at 100 kbps.

For a first test you can use the `i2cdetect` tool from a terminal window to obtain an overview of the slave devices that the system recognizes on I²C bus number 1. The results might appear as shown in **Figure 2**, where the LM75 is responding to address 0x48.

The commands `i2cget`, `i2cset`, and `i2cdump` can be used to communicate with the LM75 without having to get involved in programming. In the example below 0x00 is the register number, which must always be specified.

```
pi@raspberrypi ~ $ i2cget -y 1 0x48 0x00 w
0xa010
```

The two bytes of the reply must be swapped over, giving 0x10a0. Of this result only the upper nine bits are valid: they are 0x021. The LM75 reports temperature in steps of 0.5 K, and so the temperature reading in this example is 16.5 °C. The hardware of the Raspberry Pi in principle also supports operation as an I²C slave, but this is not supported by the Linux driver.

Programming in C and Python

Five functions are required to program the I²C bus in C.

- `open()` to access the I²C device node;
- `ioctl()` to set the I²C slave parameters;
- `read()` and `write()` for the actual communication with the slave; and
- `close()` to indicate when access to the I²C device node is no longer required.

For the following code, in addition to the other include files, the include file `linux/i2c-dev.h` is required for the definition of `I2C_SLAVE`.

```
# include <linux/i2c-dev.h>
```

The device node is accessed using the `open()` function.

```
fd = open("/dev/i2c-1", O_RDWR);
```

Next we set the slave address with an `ioctl()` call.

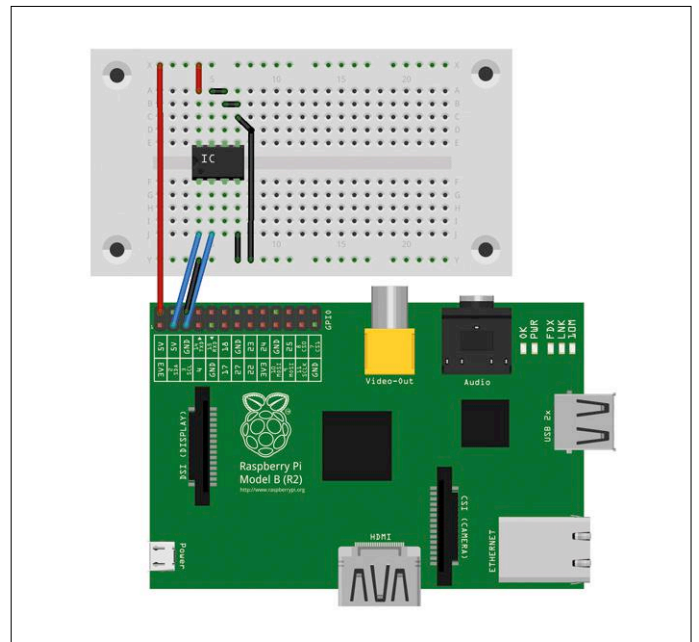


Figure 1. Connecting an LM75 to a Raspberry Pi using a breadboard.

```
ioctl(fd, I2C_SLAVE, 0x48);
```

And now we can write to and read from the device.

```
unsigned char buf[2];
float T;
buf[0] = 0;
write(fd, buf, 1); /* write register number 0 */
read(fd, buf, 2); /* read temperature register */
T = ((buf[0]<< 8) | buf[1]) / 256.0;
```

Finally we close the device node: Geräteknoten wieder:

```
close(fd);
```

Of course it is a good idea to check the return values from the function calls properly in order to detect possible errors, such

```
pi@raspberrypi ~ $ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  48  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

Figure 2. The `i2cdetect` tool has found an LM75 at address 0x48.

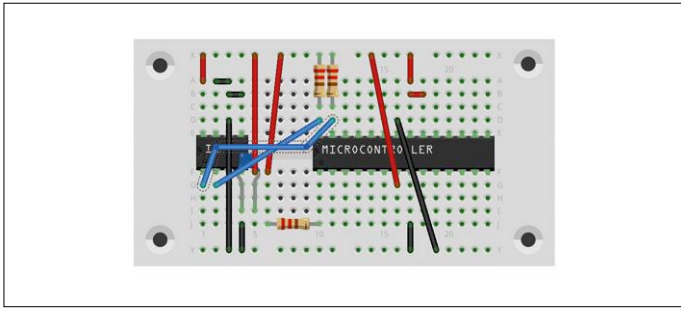


Figure 3. The ATmega88 and the sensor on a breadboard.

as whether the user running the code has permission to open the device node `/dev/i2c-1`, whether a slave exists at address `0x48`, and whether the data transfers were successful.

Before programming the I²C bus **in Python**, it is necessary to install (or to have already installed) the `python-smbus` Raspbian package.

```
sudo apt-get install python-smbus
```

Then the temperature can be read from the LM75 as follows.

```
#!/usr/bin/python
import smbus
import time
```

Listing 1. Main loop for reading from the LM75.

```
# include <i2cmaster.h>

# define LM75 (0x48 << 1)
// see datasheet

int
main(void)
{
  unsigned char val[2];

  i2c_init();
// initialisation I2C

  i2c_start(LM75 | I2C_WRITE);
// addressing to write

  i2c_write(0x00);
// temperature register

  i2c_rep_start(LM75 | I2C_READ);
// addressing to read

  val[0] = i2c_read();
// degrees Celsius

  val[1] = i2c_read();
// tenth part bit

  i2c_stop();
// ready

  for(;;);
}
```

```
bus = smbus.SMBus(1)
address = 0x48

w = bus.read_word_data(address, 0)

print format(w, '04x')
```

The “0” in the call to the `read_word_data` method supplies the register number, which here is the index of the temperature register. In the SMBus protocol, which sits on top of the I²C protocol, a register number must always be set at the start of a communication. This causes problems in the case of the PCF8574 port expansion chip, for example, which has no address register.

The returned value has the same problem as with the `i2cget` command: the two bytes in the word `w` must be swapped over.

ATmega

The Atmel ATmega microcontroller series have an integrated I²C controller which supports both standard mode (at 100 kHz) and fast mode (at 400 kHz). It can be operated as a master, as a slave or as a combination of the two. The ATmega324PB and the ATmega328PB devices include two I²C buses. In the following examples we will be using the ATmega88, which can be plugged into a breadboard along with the sensor as shown in **Figure 3**.

Here the best approach is to use the `i2cmaster` library by Peter Fleury [1]. Note, however, that this library does not automatically enable the internal pull-up resistors! It is therefore necessary to take care of this ourselves: Figure 3 shows the resistors towards the top of the breadboard, running to the 5 V rail. The library configures the I²C bus controller in the ATmega to run in standard mode (100 kHz). The LM75 sensor can be accessed using code like that shown in **Listing 1**.

Using `lcdlibrary`, by the same author, we can construct a digital thermometer with an external liquid crystal display. This can be connected using an I²C interface board (which will usually be based on a PCF8574), or alternatively an LCD with built-in I²C interface can be used.

The I²C bus controller in the ATmega devices does not have to be operated in master mode, controlling a slave. It can also be run in slave mode, controlled by an external master. A composite mode of operation is also available, where, for example, the ATmega might at one moment be communicating with the LM75 as a bus master, and then at the next moment be acting as a slave to a Raspberry Pi master. One situation where an ATmega might be used as a slave is where the ATmega is reading data in real time over its port pins, doing some processing, and then supplying the results upon request to a Raspberry Pi. This kind of set-up is more complex, and entails tight coupling with the rest of the code running on the ATmega. There do exist, however, the rudiments of a library implementing operation in slave mode [2].

The description below follows that given in Atmel’s datasheets, which contain tables of the various states of the I²C controller. For example, in the datasheet for the ATmega48/88/168, the relevant information can be found in section 22.7.

It is sensible to make the software implementing I²C slave mode run under interrupts. The bus controller hardware can trigger an interrupt under the following conditions.

- after transmitting a 'start condition' or 'repeated start condition';
- after transmitting the address and read/write bit;
- after transmitting a data byte;
- when the ATmega has lost an address arbitration (when a collision occurs while transmitting the 'start condition', the address byte or the read/write bit);
- when the ATmega has detected a 'start condition' and has been addressed as a slave;
- when the ATmega has received a data byte;
- when the ATmega has detected a 'stop condition', or a 'repeated start condition' where it has (again) been addressed as a slave; or
- when an invalid bus transaction has been detected.

The first four of these situations are only relevant to master mode, and so we are only interested in the last four. The I²C peripheral unit must be initialized with the desired slave address. It can then be enabled and it will start to run.

```
TWAR = (I2C_Slave_Addr << 1);
TWCR = _BM(TWEA) | _BM(TWEN) |
_BM(TWIE);
```

There is no need to set the bit rate for slave mode, as the communication speed is determined by the master. When an interrupt occurs, the first step is to determine its cause. To do this it is necessary to read the TWSR status register and examine its top five bits.

Listing 2 shows a fragment of the interrupt service routine (ISR) for receiving and transmitting data as a slave, and for detecting a 'stop condition'.

The descriptions and tables in the ATmega datasheets are very comprehensive, including explanations of the status codes, actions required in software, and the resulting behavior of the hardware.

Arduino

Many Arduinos are based on Atmel ATmega-series microcontrollers, and so the above discussion applies equally to them. However, a popular and very convenient library called [Wire.h](#) is also available for the Arduino.

Working with the Arduino is just as straightforward as working with the Raspberry Pi (see **Figure 4**). Depending on the exact model of Arduino (or clone), you may find that the processor

Listing 2. Interrupt service routine to deal with address matching and detection of 'stop condition'. The complete and extensively commented code is available on the project web page [5].

```
ISR(TWI_vect)
{
    /*
     * These variables need to be preserved across interrupts
     */
    static unsigned char i2c_idx,      /* Index into twi_msg[] */
                       i2c_tosend;    /* Number of bytes to send */

    switch (TWSR & 0xf8)
    {
        ...
        /*
         * RECEIVE Code
         * See Table 19-4. Status Codes for Slave Receiver Mode
         * [Page 229]
         */
        case 0x60:
            /*
             * Own SLA+W has been received; ACK has been returned
             * TWDR: No TWDR action
             * STA=X STO=0 TWINT=1 TWEA=1
             * Data byte will be received and ACK will be returned
             */
            TWCR = (TWCR & ~_BM(TWSTO)) | (_BM(TWINT) | _BM(TWEA));
            i2c_idx = 0;
            break;

            ...

        case 0xA0:
            /*
             * A STOP condition or repeated START condition has been
             * received while still addressed as slave
             * TWDR: No action
             * TWA=0 STO=0 TWINT=1 TWEA=1
             * Switched to the not addressed Slave mode;
             * own SLA will be recognized;
             * GCA will be recognized if TWGCE = "1"
             */
            TWCR = (TWCR & ~(_BM(TWSTA) | _BM(TWSTO))) | (_BM(TWINT) | _BM(TWEA));
            break;

            ...

        case 0xA8:
            /* Own SLA+R has been received; ACK has been returned
             * TWDR: Load data byte
             */
            /*
             * The address (register number) has been received,
             * Start sending payload
             */
            TWDR = 0x42;
            break;
    }
}
```

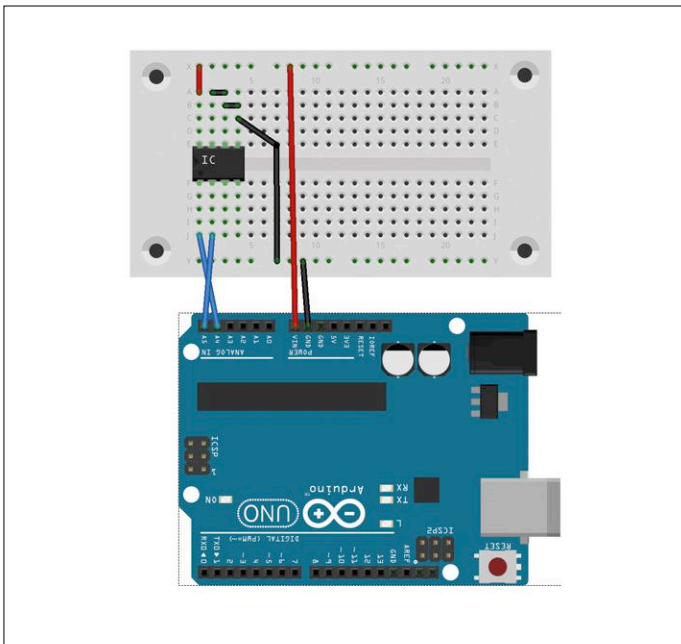


Figure 4. Connecting to an Arduino is practically the same as connecting to a Raspberry Pi.

runs on 3.3 V and that its ports cannot tolerate 5 V signals. So check and measure before connecting your circuit and if necessary remove any pull-up resistors on the slave.

The `Wire` library can be found directly within the Arduino IDE. As in the case of Peter Fleury's `i2cmaster` library the library and I²C interface must be initialized at the start of your code.

```
#include <Wire.h>
void setup() {
  Wire.begin();
}
```

The call to `Wire.begin()` enables the internal pull-up resistors. The value of these resistors is relatively high, which can cause problems: if so, add two external resistors with a value of 10 k Ω to 20 k Ω in parallel. Alternatively, disable the internal resistors altogether and just rely on external pull-ups (which should then be in the region of 4.7 k Ω). This can be done with the following two lines of code after the call to `Wire.begin()`.

```
digitalWrite(SDA, 0);
digitalWrite(SCL, 0);
```

The I²C bus on a PC

Practically every PC has its own I²C interface, and most have several such interfaces. There are I²C slaves in displays (DDC [3]) and DRAM DIMMs (SPD [4]). Internal temperature sensors are also often connected over I²C. Unfortunately there is scant to non-existent manufacturer information on the devices used and on whether and how the buses can be accessed externally: sometimes a bus will be used only within a particular module. If the PC is running Linux, it is easy to run some experiments by loading the `i2c-dev` module as follows.

```
$ sudo modprobe i2c-dev
```

You can then look at what device nodes are present in `/dev`.

```
$ ls /dev/i2c*
/dev/i2c-0 /dev/i2c-1 /dev/i2c-2 /
dev/i2c-3 /dev/i2c-4 /dev/i2c-5
```

Up to this point you do not need root privileges. Under Debian and its derivatives (which includes Raspbian and Ubuntu) you can install `i2c-tools`.

```
apt-get install i2c-tools
```

Sometimes there are slaves on only one of the buses, as in the following example.

```
root@bounty:~# i2cdetect -y 5
0 1 2 3 4 5 6 7 8 9 a b c d e f
00:  ---
```

```
10:  ---
20:  ---
30:  -- -- -- -- -- 37 -- -- 3a -- -- --
40:  ---
50:  50 -- -- -- -- -- 58 -- -- -- -- --
60:  ---
70:  ---
```

It is not always clear what peripheral devices are present. In the example below the monitor's EDID PROM is at address 0x50.

```
root@bounty:~# i2cdump -y 5 0x50
No size specified (using byte-data access)
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  0123456789abcdef
00: 00 ff ff ff ff ff ff 00 1a b3 d4 07 ec 22 02 00 .....?????"?.
10: 0a 16 01 03 80 34 20 78 2a ef 95 a3 54 4c 9b 26 ?????4 x*??TL?&
20: 0f 50 54 a5 4b 00 81 80 81 00 81 0f 95 00 95 0f ?PT?K.???.???.??
30: a9 40 b3 00 01 01 28 3c 80 a0 70 b0 23 40 30 20 ?@?.??(<??p?#@@
40: 36 00 06 44 21 00 00 1a 00 00 00 fd 00 38 4c 1e 6.?D!..?..?.8L?
50: 52 10 00 0a 20 20 20 20 20 20 00 00 00 00 fc 00 42 R?.? ...?.B
60: 32 34 57 2d 35 20 45 43 4f 0a 20 20 00 00 00 ff 24W-5 ECO? ....
70: 00 59 56 32 45 31 34 30 30 31 32 0a 20 20 00 8e .YV2E140012? .?
```

If your PC has a VGA connector, you can try connecting an I²C slave to pin 12 (SDA) and pin 15 (SCL). Pins 6 (SCL) and 7 (SDA) of a DVI connector and pins 15 (SCL) and 16 (SDA) of an HDMI connector are also good candidates for experimentation. Unfortunately in some cases these buses are under the sole control of the graphics card and its firmware.

Note that this is not an official solution and that things may change in the future. In any case it is a good idea to measure the effective pull-up resistance with a multimeter after initialization is complete.

Caution: when the internal pull-up resistors are disabled in this way there will be a brief period during which they are enabled. This can cause problems if a 3.3 V slave with two external pull-ups to 3.3 V is connected to a 5 V Arduino. In this case it is essential to use a level-shifting circuit.

After initialization, data can be sent to the I²C slave in the `loop()` function. The following example sets the address pointer in the LM75 to point to the temperature register.

```
void loop() {
  Wire.beginTransmission(0x48);
  Wire.write(byte(0x00));
  Wire.endTransmission();
}
```

The following fragment reads from the LM75.

```
Wire.requestFrom(0x48, 2);
  c1 = Wire.read();
  c2 = Wire.read();
}
```

As you can see, the `Wire` library requires that you first specify the number of bytes expected from the slave (in this case, 2). The first call reads in the two bytes, and then the subsequent calls allow you to access the received data.

The Arduino `Wire` library can also be configured for operation in slave mode. Again, the library must first be initialized. In this case the call to `Wire.begin()` must be given a parameter which is the desired slave address: it is the presence of this parameter that selects slave mode. It is also necessary to set up an event handler which will be called whenever the Arduino is addressed as a slave. In the example below the event handler is called when the Arduino is to receive data.

```
#include <Wire.h>
void setup() {
  Wire.begin(0x48);
  Wire.onReceive(receive);
}
```

Although not normally necessary for a slave device, the call to the `Wire.begin()` method again enables the built-in pull-up resistors. They can be disabled if necessary as described above.

Caution: note again that although a 5 V Arduino can be configured as a slave to a 3.3 V master, there will be a brief pulse to 5 V on the signal lines which may damage the 3.3 V master. So, for example, if you want to operate an Arduino as a slave to a Raspberry Pi, you must make certain that the Arduino is only electrically connected to the Raspberry Pi after it has completed initialization.

When a data byte is received, the event handler (which was configured to be `receive` in the above example code) will be called. Again the data bytes have already been read in, and the number of bytes received is passed as a parameter to the handler.

```
void receive(int n) {
  while (n-- > 0) {
    uint8_t c = Wire.read();
    // Verarbeite c
  }
}
```

If you wish to transfer data to the master then you should call the method `onRequest()` instead of `onReceive()`. Again, the name of the event handler function that will produce the data to be sent is passed as a parameter. Since at the moment of calling the handler function the number of bytes to be sent is not known, no parameter is passed to it. The data bytes are sent using **a single** call to `Wire.write()`.

```
#include <Wire.h>
void setup() {
  Wire.begin(0x48);
  Wire.onRequest(transmit);
}
void loop() {
  while (1) delay(1000);
}
void transmit() {
  uint8_t msg[N];
  // Erzeuge msg[] Inhalt
  Wire.write(msg, N);
}
```

The two event handler set-up methods can be called together in the same sketch.

```
#include <Wire.h>
void setup() {
  Wire.begin(0x48);
  Wire.onReceive(receive);
  Wire.onRequest(transmit);
}
```

Such a configuration would allow you to emulate an LM75 accurately, using a 1-wire temperature sensor such as the DS18B20 instead of the LM75, or to emulate a real-time clock peripheral receiving time over DCF77 or GPS.

Coming up in part three

The next installment in this short series will look at some popular I²C peripheral devices: besides the LM75 temperature sensor we will also examine the PCF8574 port expander and the RV-8523 real-time clock. Finally we will close with some thoughts on troubleshooting using tools within the means of the average hobbyist. ◀

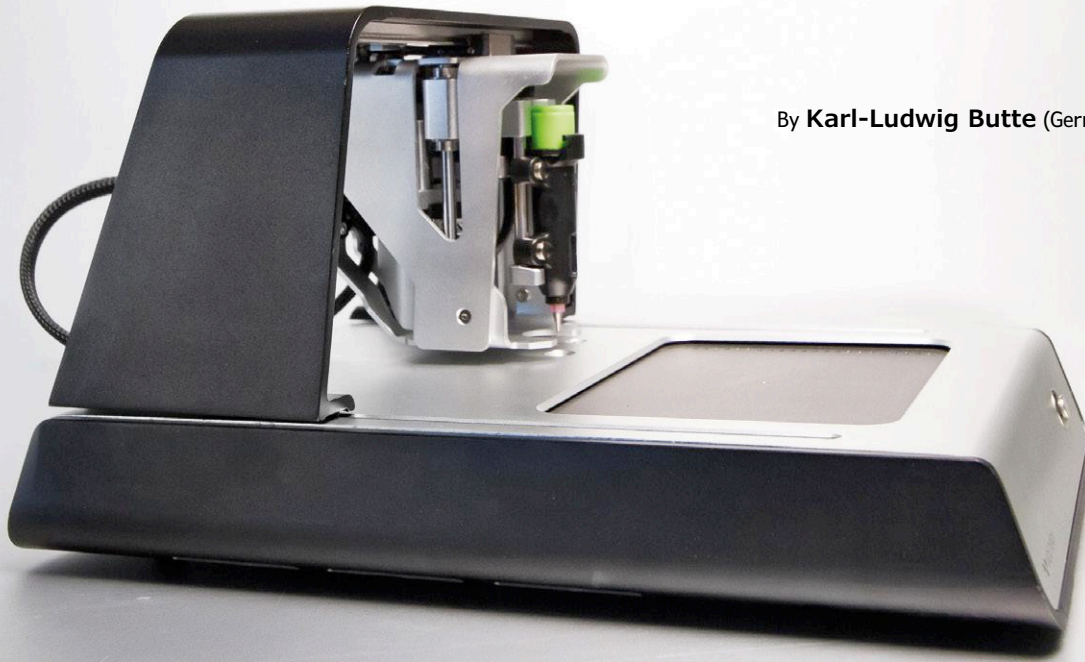
(160418)

Web Links

- [1] <http://homepage.hispeed.ch/peterfleury/avr-software.html>
- [2] www.jtronics.de/avr-projekte/library-i2c-twi-slave.html (in German, English machine translation available)
- [3] https://en.wikipedia.org/wiki/Display_Data_Channel
- [4] https://en.wikipedia.org/wiki/Serial_presence_detect
- [5] www.elektormagazine.com/160148

PCBs — Printed, not Etched

Presenting the Voltera V-One PCB printer and reflow soldering station



By **Karl-Ludwig Butte** (Germany)

Figure 1. Side view of the Voltera V-One.

Etching your own PCBs used to be a real chem lab exercise. First you had to copy the PCB layout from film to the PCB material by exposing, developing and fixing the photoresist. That involved a lot of chemicals, with unpleasant odors and a good chance of splashes or spills. Now there's an entirely new option: the Voltera V-One can print circuit tracks, apply solder paste, and even solder the components in a reflow process.

The hassle of working with chemicals convinced many developers to stop making their own prototype PCBs. The trouble starts already with the layout film, where you have to determine the exposure and development times by trial and error. The goal is to find the best compromise between the darkness of the black tracks and the lightness of the surrounding areas. Making the tracks darker causes the surrounding areas to become gray, with the result that the copper will not be fully etched away where it shouldn't be left. If instead the surrounding areas are nicely clear, the track density is not high enough and thin tracks in particular are sometimes etched away.

Once you get the film prepared reasonably well, it is time for the real work: exposing, developing and etching the PCB. If everything goes well, after removing the photoresist layer you are rewarded with shiny copper tracks. We call this a printed circuit board (PCB), but actually it should be called an etched circuit board (ECB).

Now the Canadian start-up Voltera is poised to revolutionize

PCB production with their V-One PCB printer, which could inspire many electronics enthusiasts to start making their own PCBs. That's true twenty-first century technology — with the V-One you print PCBs, apply solder paste to the board, and even solder manually placed components using a reflow process.

Hardware

The V-One is a flatbed printer with three-axis control, and it fits easily on a benchtop (**Figure 1**). The mechanical structure is illustrated in **Figure 2**. There is a base on which a movable bridge is mounted for the Y axis. A carriage with a tool holder for three different tool modules and a drive gear (carriage gear) is mounted on this bridge. This carriage travels along the X axis. The tool modules can also be moved vertically along the Z axis. The three tool modules shown in **Figure 3** are a probe head, a print head for electrically conductive ink, and a print head for solder paste. The tool modules are simply attached to the tool holder by four strong magnets, making them easy to exchange.

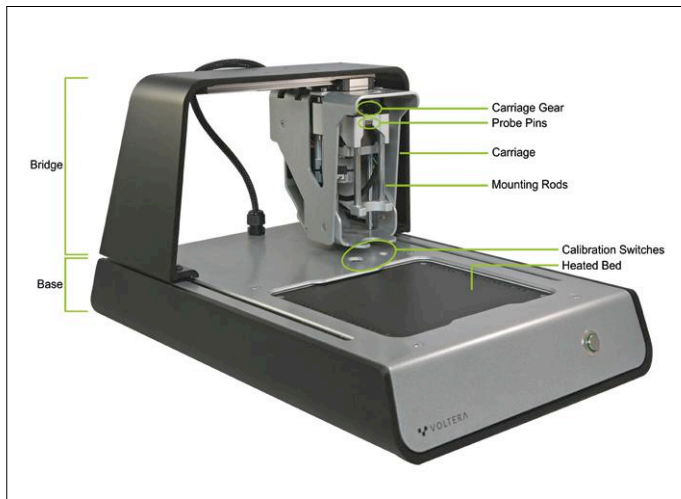


Figure 2. Mechanical structure of the V-One.



Figure 3. Tool modules of the Voltera V-One.

The PCB substrate material for printing (FR4) is secured to the work surface by two clamp strips. A heating system is mounted beneath the work surface. In addition to heat-setting the printed tracks, it allows SMD components mounted on the PCB to be reflow soldered in a subsequent operation. This means that with the V-One you can not only make your own PCBs, but also solder components on your boards, eliminating the need for separate reflow oven. And because the V-One also applies solder paste precisely on the pads, you do not need a stencil. With the V-One, every electronics enthusiast can work with SMD components easily and professionally.

The slots which guide the bridge along the Y axis are illuminated by recessed strips of RGB LEDs (**Figure 4**). In addition to being a visual attraction, they signal the temperature of the work surface: red means "Hot! Don't touch!" In that state the work surface temperature can be as high as 250°C. When the LEDs are blue, the thermal phase of baking or reflow soldering has been completed and the board can be removed.

The printer has a USB interface, and at least for operation with Windows a specific driver must be downloaded and installed from the Voltera website.

Software

The associated software is a package comprising calibration, control and printing software and a video user guide. The video user guide in particular deserve praise: each step of the procedure is illustrated by one or more short videos on the right side of the screen, and each action is shown in practice (**Figure 5**). The videos run continuously and help you understand the illustrated actions on your own machine. It could hardly be simpler or clearer.

Software is available for Microsoft Windows 7 (64-bit) or later and for OS X version 10.11 or later. Unfortunately, a Linux version is not yet available. The software can be updated at any time, with no need for antiquated optical storage media. In addition, the software checks for updates each time the program is launched. If an update is available for the software or the printer firmware, it is installed automatically.

Working with the V-One

The Voltera V-One processes exclusively Gerber files, which can be generated by most modern PCB layout programs. Volt-

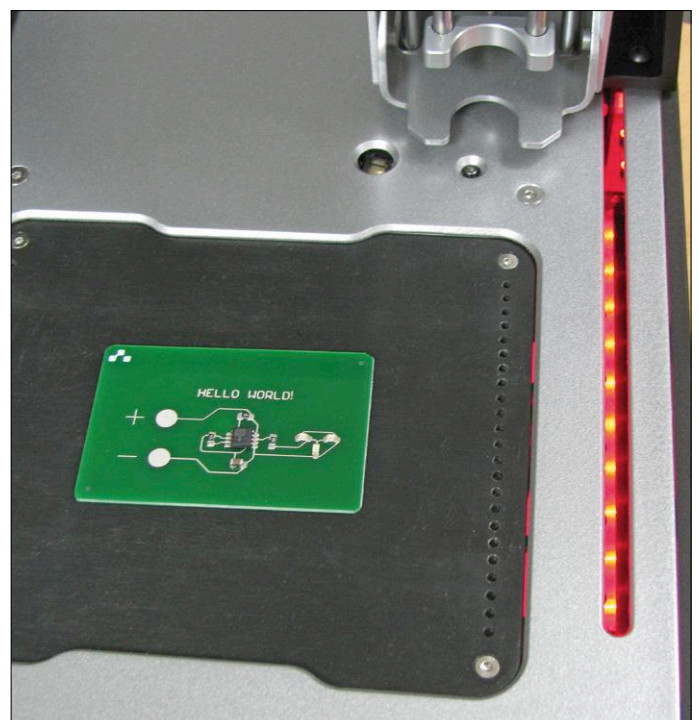


Figure 4. Operating mode indication: when the LEDs in the Y axis slots are red, the work surface is very hot.

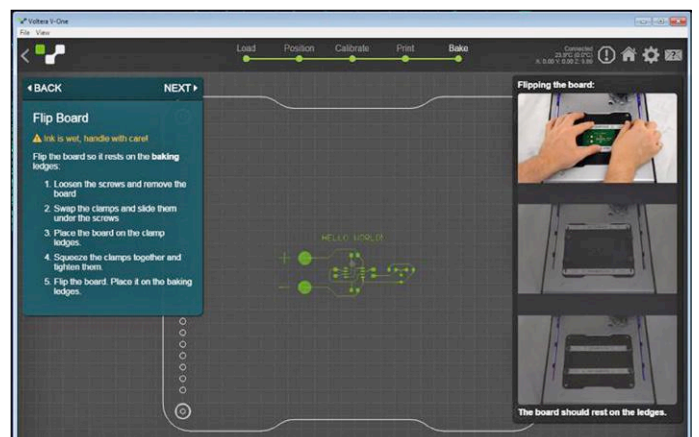


Figure 5. The software package for the V-One includes video tutorials.

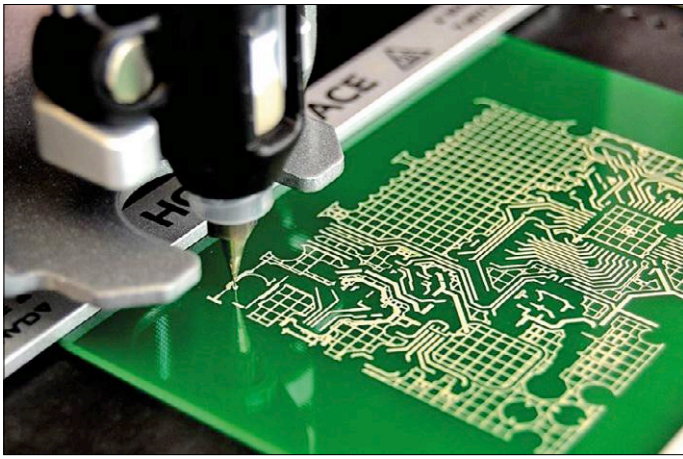


Figure 6. The V-One at work: printed PCB tracks.

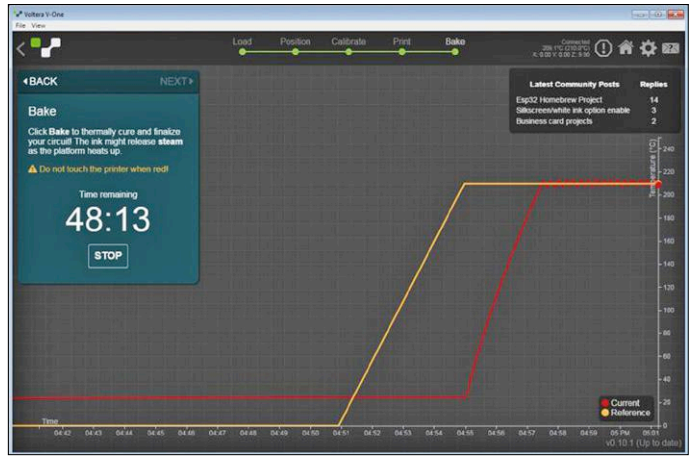


Figure 7. Target and actual temperature curves for PCB track baking.

era has tested the Gerber dialects of Eagle, Altium, Upverter and KiCad. Gerber files are also available as downloads for many Elektor projects. To make it easy to try out the printer right away, Voltera kindly includes all necessary components for a small “Hello World” project. That consists of a simple blinking light board with the well-known NE555 timer IC in an SMD package. The PCB layout is already incorporated in the software and can be accessed by a special link in the Open File dialog.

Printing

Standard FR4 glass fiber reinforced epoxy circuit board material is a suitable choice for the substrate. Along with a ten-pack of blank circuit boards with dimensions of 2 × 3 inches and 3 × 4 inches, my V-One came with a shield for the Arduino Uno and Mega. The maximum printing area is 138 × 102 mm (5.5 × 4 inch). In my experience it is a good idea to clean the board with alcohol before printing, since traces of skin oil can lead to broken PCB tracks.

The first step is to secure the blank circuit board in the middle of the working area with the two metal clamp strips. In the next step, the probe head outlines a rectangle on the board within which the circuit will be printed. If the rectangle is not in the middle of the board or extends beyond the edge of the board, you can adjust the position on the screen until everything is okay. You can also use this approach to print several small layouts on the same board, for later separation into individual boards.

Once the position on the board is correct, the probe head scans the board to generate a height profile. The print nozzles for the electrically conductive ink and the solder paste are very thin and fragile. They must never be allowed to collide with the board surface, but they also should not float too high above the surface, as otherwise the print quality will suffer.

After the height profile has been generated, the probe head is replaced by the print head with the electrically conductive ink. First a test pattern is printed with this head. It consists of two serpentine lines and several parallel horizontal lines with a minimum separation of 0.8 mm. The minimum track width is 8 mil (about 0.2 mm). If the test pattern is not printed properly, you can adjust the ink flow in the software and then check the new setting by printing the pattern again (after wiping off

the previous one). According to the manufacturer, the conductive ink and the solder past should normally be kept in the refrigerator, and they should be taken out 30 minutes before the start of printing to allow them to adjust to room temperature. However, in my experience a considerably longer time of about two hours is advisable, because then the printing is much more uniform and above all free from gaps. If the test pattern is okay, you can wipe it away with a paper towel and clean the board again with alcohol.

It is always fascinating to watch the printer at work (**Figure 6**). Unlike ink jet printers, the print head does not scan back and forth over the surface of the board. Instead, it prints the tracks one after the other, just like a plotter, and finishes them with solder pads.

▶ Automatic PCB printing and reflow soldering

If the printing is not satisfactory in some places, which can happen for a wide variety of reasons, you can select the affected area on the screen and repeat the printing in that areas. For this you can also adjust the ink supply if necessary.

Baking

Once the printing is completed, you can start the baking process in order to cure (harden) the freshly applied ink tracks. Until they are cured, you must be careful to avoid accidentally smearing the ink. For baking, the clamp strips are exchanged and the board is placed in the recesses of the clamp strips with the printed side facing down, so that the surface of the board is held slightly above the working surface.

The baking process is initiated by the software and takes about 40 to 50 minutes. During this time the target and actual temperatures of the heating system are continuously plotted on the screen (**Figure 7**). The bridge guide slots also light up bright red as a warning signal. Touching the printer during the baking process is not a good idea, because a temperature of 250°C can cause painful burns. To on the safe side, I placed my V-One on top of a double layer of ceramic tiles.

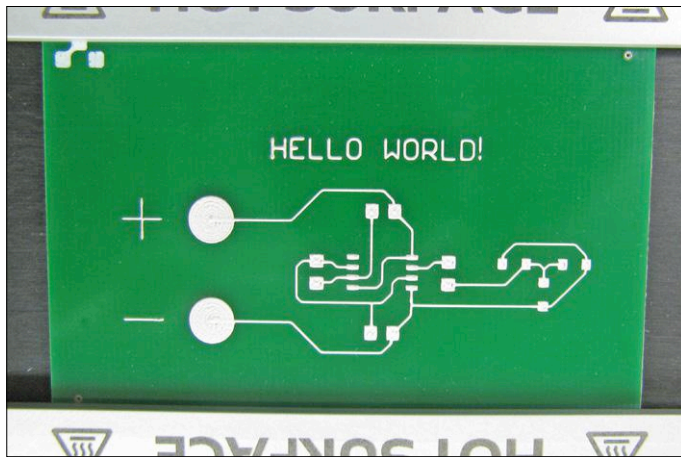


Figure 8. The fully printed and polished PCB after track baking.

When the LEDs change back to blue, the heating system has cooled down and the baking process is finished. Now you can safely remove the board.

A special plastic sponge is included with the V-One, and it must be used now to polish the PCB in order to ensure reliable solder joints later on. **Figure 8** shows the fully printed and polished PCB.

Applying solder paste

Before starting with the application of solder paste, you have to mount the PCB back on the working surface, and then you must tell the printer the precise locations of two reference points. That is easily done with the probe head in combination with the appropriate software functions. Then you replace the probe head by the print head for solder paste and start the printing process.

Component placement and reflow soldering

Placing the components on the board is a manual task. Good tweezers are essential for this. Even so, you must be very careful to avoid smearing any of the solder paste, and you must ensure that polarized components such as electrolytic capacitors, diodes, ICs and so on are placed with the right orientation. When all the components are mounted, you can place the board in the middle of the working area and start the reflow process (**Figure 4**). When the LEDs change back to blue, the board is finished and ready for testing (**Figure 9**). The connecting wires for the 9 V battery clip (on the left in **Figure 9**) are soldered on afterwards, which brings us to the next topic in this article.

Processing leaded components

In response to the question of whether the V-One can only process SMD components, there is a clear answer: it can also be used with leaded components. For this you must use the corresponding solder pads (with a hole in the middle) in the layout. After the board is baked and polished, the next step is old-fashioned manual hole drilling and component placement, instead of solder paste application. However, electrically conductive ink does not behave the same way as copper, so you have to get used to soldering on it. It is essential to use a temperature-controlled soldering station which can be set to

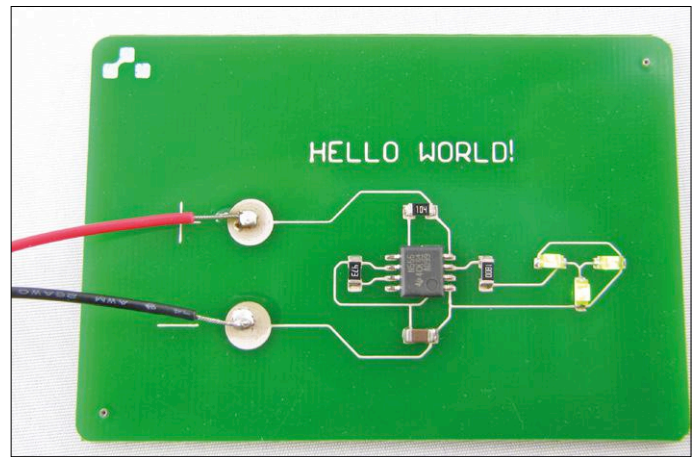


Figure 9. After soldering by the V-One, the finished board is ready for testing.

temperatures in the range of 190 to 210 °C. The solder recommended by Voltera (Sn62, or even better SnBiAg1) is well suited to this temperature range. The solder wire is so thin that there is no room for a flux core. The flux (for example, ChipQuick SMD 291) must be applied separately before soldering. As I found out the hard way, that takes a bit of practice.

Summary

The Voltera V-One definitively makes PCB etching a thing of the past. Once you have tried it, you can hardly imagine life without it as an electronics enthusiast. Particularly for hardware development, it is a major advantage to not have to wait days or weeks for prototype PCBs from a service provider. The integrated reflow soldering capability is the icing on the cake and makes SMD soldering a routine task. Although the PCB printer is not exactly cheap (the Voltera list price is \$2179), and neither are the associated consumables, in my opinion they are all worth every penny. ◀

(160384-1)

Web Link

[1] www.voltera.io

About the Author

Karl-Ludwig Butte studied Computer Science with specialization in economics at the Fulda University of Applied Sciences in Germany and is presently active in IT and process control in the automotive industry. In his free time he enjoys working on electronics projects, as well as computers of all types and the history of computers.

nWatch, a Wearable Development System or how to build your own smart watch



By **Piotrek Wasilewski** (Poland)

Originally designed as a simple school-quiz-electronic-cheat-sheet, this watch became pretty smart over time. And by evolving it into a more useful device the author accidentally learned a few things along the way. So, cool gadget or fun educational tool? Build one yourself and find out.

The nWatch presented in this article is a smart watch that doubles as a microcontroller development board. At the start this project wasn't really about a development board, but more for a simple watch to cheat at my school quizzes about subjects that were — in my humble opinion — not relevant. The first prototype was not very advanced, but it did what it was designed for. Then I started thinking about the possibilities of developing a device that would be useful and allow me to learn something in the process, and so I came up with the idea to build a smart watch that doubles as a microcontroller development board. This turned out to be not so easy though. When I started out I had limited knowledge of electronics and only some basic programming skills, the idea being that I would acquire the necessary knowledge along the way. The goal was to make the watch from parts accessible to enthusiasts like me, available in shops or through the Internet. In this article I will show you how I approached this project and prove that anyone can build a similar device at home.

Always start @ the beginning

One day I woke up with the idea of building a tactile smart watch with a big LCD and some other features like an MP3 player, SD card, etc. But what to do with such an idea? It is generally good practice to write down the things you want to do, the pros and cons of the solutions you plan to adopt, the functions of the device, and so on. In short, it all starts by defining the specifications of the device. At first I thought I could skip this step and just draw up a schematic and design a PCB for it. However, as it turned out this first, naive version was so riddled with mistakes that I was forced to make a second one and then a third. Then I

finally realized what I was doing wrong: I wanted to make a quite complex device in a very short time. This really was a bad idea and so I decided to spend some time to rethink the schematic, and redo the PCB while aiming for as few bugs as possible. After about two weeks of fixing issues, I came up with the final version. I had learned to think about a project first before trying to draw the schematic or design a PCB. This project also taught me to be patient.

Features

- ARM Cortex-M4
- Bluetooth Low Energy (BLE)
- Gyro, accelerometer and magnetometer
- MP3 player
- 3D printed enclosure
- Great learn project

is a member of the high-performance foundation line from STMicroelectronics. It has an ARM Cortex-M4 core running at 168 MHz with a DSP and an FPU, there is 1 MB of flash memory, and a galaxy of peripherals. The reasons for selecting this device are several. First, it has enough general purpose I/O (GPIO) ports and buses to connect all the other ICs to it. A special hardware memory management unit (MMU) is present, which is used to control the LCD and

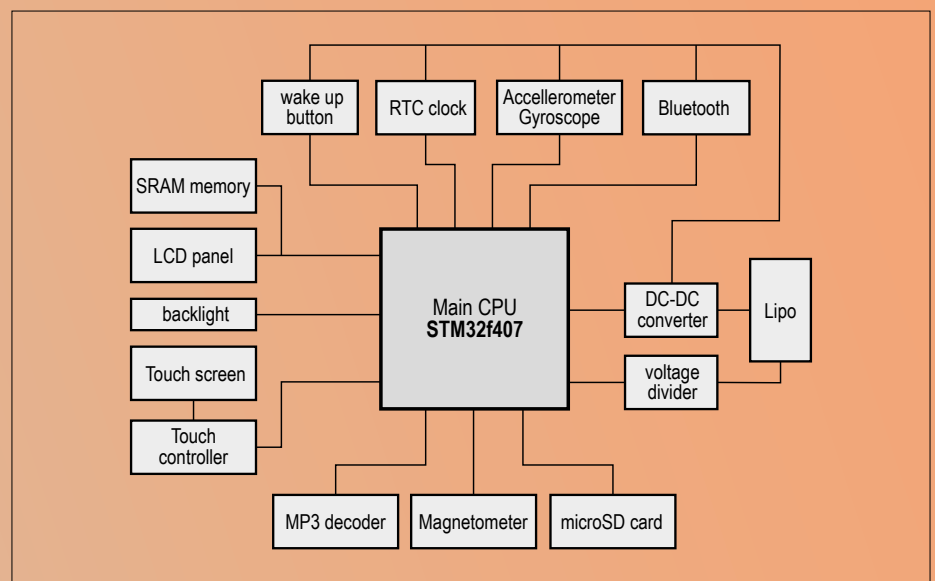


Figure 1. Overview of the nWatch internals. It was not easy to cram all this into a wristwatch.

The nWatch hardware

Figure 1 shows the block diagram of the nWatch, the schematic that closely follows the block diagram is shown in **Figure 2**. As you can see there are quite a few blocks and I will discuss them one by one in no particular order.

The CPU

The heart of the nWatch is IC1, an STM-32f407ZGT6 microcontroller. This micro

the external SRAM memory (IC10). The MMU makes controlling things much easier than would be possible with independent GPIOs pins. The micro has fast built-in RAM, and enough flash memory for my purposes. Using this micro also has one major drawback: its case, which is pretty huge (relatively speaking, of course). Unfortunately there was no other solution, because I didn't want to use a ball-grid array (BGA) package.

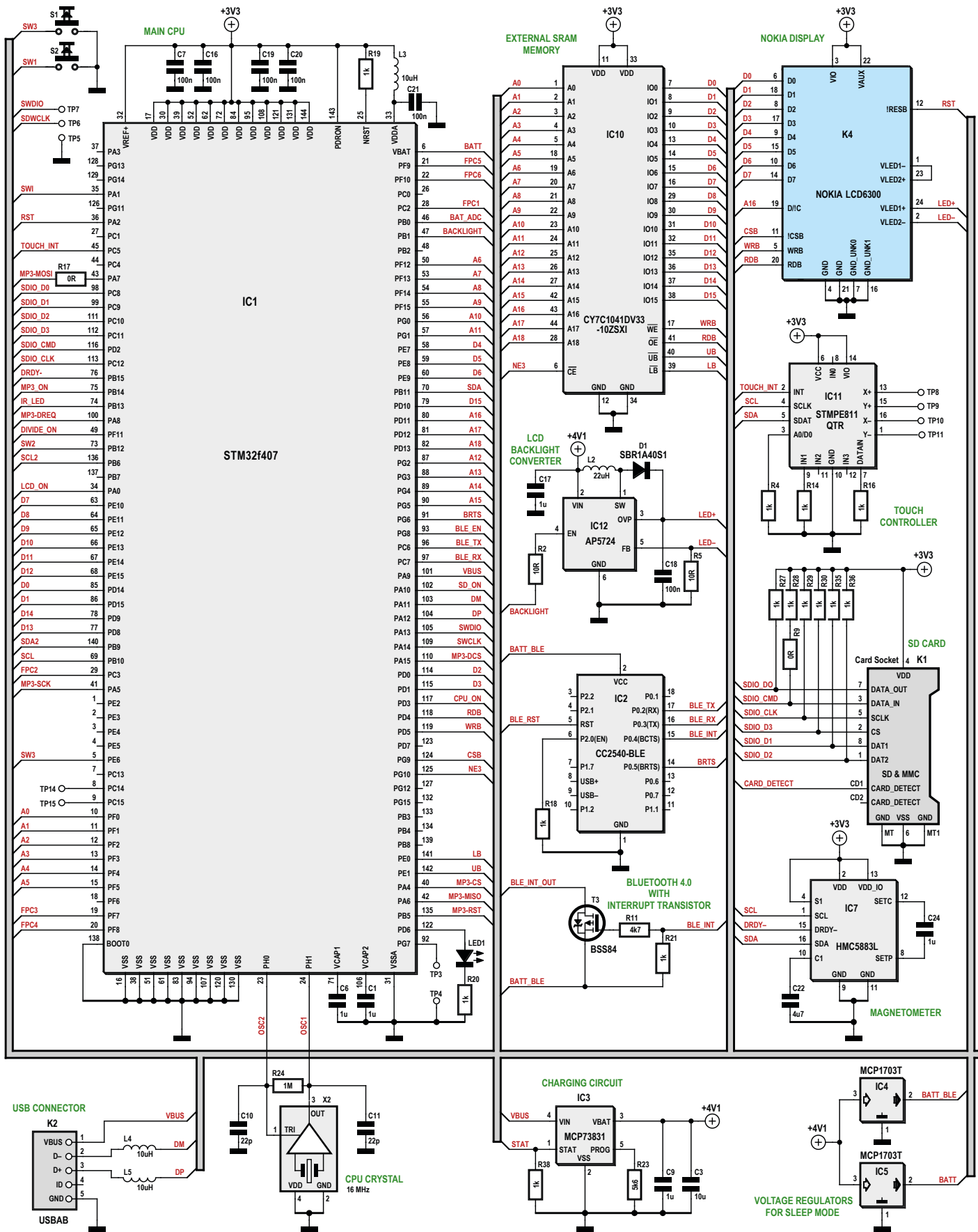
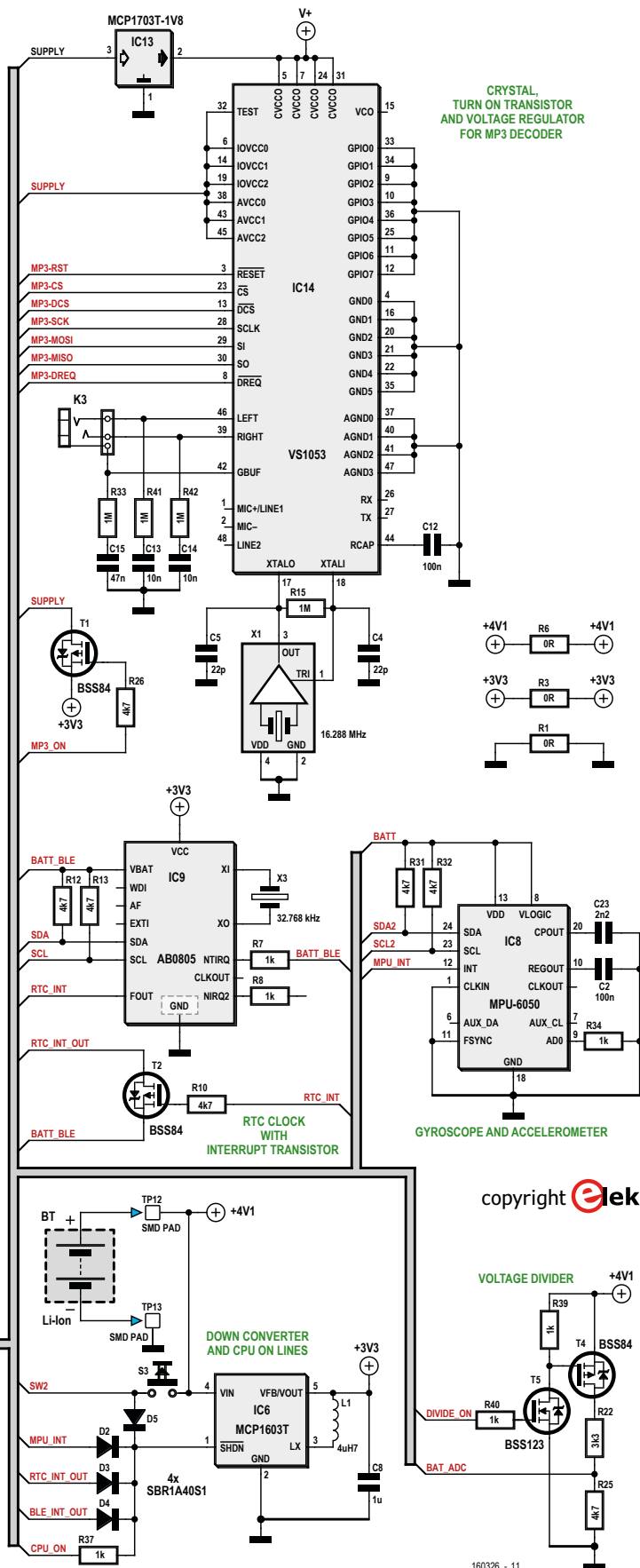


Figure 2. Besides some resistors and capacitors, the schematic does not add much to the block diagram.



copyright elektor

External SRAM and microSD card

There are two external memories in the nWatch, an SRAM chip and an SD card. The SRAM IC10 was not originally planned, but it eventually proved necessary to handle the display in a comfortable way because the CPU's internal RAM is not large enough to handle the memory requirements of the STemWin graphics library (see below). IC10 is used as a screen buffer for the LCD. All screen modifications are applied to the buffer held in this memory before it is transferred to the display's buffer. Notice that this memory is not as fast as the micro's built-in RAM, and it is good practice to keep all the allocated stacks (FreeRTOS memory, etc.) in internal memory to make the CPU work as fast as possible. The external SRAM is connected to the same bus as the LCD and is controlled by the CPU's flexible static memory controller (FSMC).

The second external memory is the non-volatile SD card (K1). I opted for an SD card because you can remove it easily and connect it to a computer or smartphone to upload a new image or text file. This part is a leftover from the first versions of the cheat watches. At the time I didn't know how to upload cheat sheets to my watch other than by using an SD card. It is connected through a fast SDIO bus which allows reading and writing files much faster than over SPI.

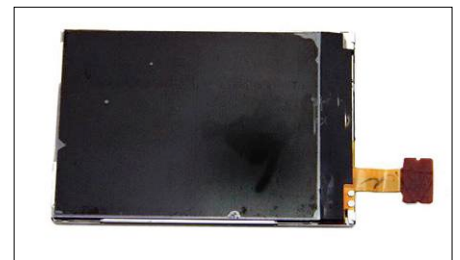


Figure 3. The Nokia 6300 display is still available on the Internet.

The display

Actually the display is the reason for the letter 'n' in 'nWatch', as it was scavenged from an old Nokia 6300 mobile phone (Figure 3). Having picked this display was no accident. It is quite easy to control as it uses an 8-bit parallel bus; it is cheap (less than \$5); and it has a connector that is not unthinkable for use by an amateur. Furthermore, it has the right size. It is small enough for a watch

and large enough to allow finger movements (and a touch panel) to control the watch. Reverse engineering the display seemed not too hard — on the Internet I found articles describing how to communicate with the display, and even the datasheet of the LCD controller ended up on my hard drive. Although these properties had convinced me to use the Nokia display for my watch, I did do a prototype with one of those popular ILI9320-based TFT LCDs. At the time I thought that the display would be much faster if I used a 16-bit bus instead of 8-bit, and also the ILI9320 controller chip has more built-in features. However, this experiment, albeit interesting, didn't satisfy me. It turned out that the LCD had poor viewing angles and I found the colors on the pale side. Maybe my TFT was not of the best quality, I don't know, but this experiment made me decide to stick to the Nokia display. OK, I know it is not the fastest gun in town, but it is good enough for simple user interfaces. Remember that the way the display is drawn also highly influences the speed. Drawing it pixel by pixel will be much slower than transferring big chunks of data using Direct Memory Access (DMA). The LCD connects to connector K4.

Adding a touch panel

Having a touch panel was one of the main objectives of the project. However, finding a touch panel that fitted my specifications was pretty difficult and, even though I succeeded, now, in hindsight, I even consider this a big drawback as it seriously compromises reproducing the nWatch.

The first panel I tried was from a 2010 Nokia C3 phone. This is a perfect match for a 2.2" N82 LCD, which is a larger and almost fully compatible brother of the Nokia 6300. Although it was resistive, its sensitivity was rather good. Unfortunately, it was too large for a wristwatch and it didn't fit the 2" Nokia 6300 display. After many hours of surfing and searching the Internet, I finally found a suitable touch screen on Aliexpress. Its size was perfect for my display, but its sensitivity was not impressive, requiring strong finger presses and taps. Although I use this panel in the current nWatch, I'm not happy with it, and I would like to replace it by a capacitive touch panel. The four touch panel wires go to TP8 and TP11.

Touch panel controller

To interface with the resistive touch panel I opted for the distinguished STMPE811 from STMicroelectronics (IC11). It can talk to the MCU over SPI or I²C (used in this case), and has many registers allowing in-depth control of the touch panel. Different parameters can be adjusted to adapt the touch screen to the display you are using. It also has some interesting hardware features like window tracking or temperature measurement. It can generate an interrupt on touch, informing the CPU that it can read the coordinates instead of doing periodic register reads.

Backlight

The AP5724 from Diodes Inc., IC12, is a small white LED step-up converter in a 6-pad DFN package that controls the brightness of the LCD. Being a step-up converter it needs a few external components like a power inductor (L2), a Schottky diode (D1), and capacitors (C17 & C18). It is controlled by a PWM signal from the CPU. It is able to drive up to six LEDs, and since the Nokia display has only two, it is perfect for the job. Resistor R5 determines the maximum current through the LEDs.

MP3 player

In the first two prototypes the main CPU was responsible for playing music files. An external digital-to-analog converter (DAC) was used to drive the earphones. Only WAV files were supported, and everything — like file reading, header parsing, etc. — was done in software. MP3 support would be nice, but since I didn't like too much the idea of decoding MP3 files myself in software, I looked for another solution. This search led me to the exquisite VS1053 from VLSI Solution (IC14), which can decode and play different file formats (Ogg Vorbis, MP3, AAC, WMA, FLAC, it even plays MIDI files) — all you have to do is send them to the chip over SPI. This chip not only does the hard work, it also provides functions to control the sound (volume and tone control, etc.), and it has an on-chip headphones amplifier. In short: the perfect solution. Thanks to IC14, all the CPU has to do is periodically send chunks of data to the decoder chip.

The VS1053 is quite small, and only needs a few capacitors, a few resistors and a quartz crystal. It requires 1.8 V for the core (C_{VDD}), which led to the addition of voltage regulator IC13 to the board.



Figure 4. Thanks to a dedicated decoder IC the nWatch can be used as MP3 player.

Transistor (T1) is used to switch the decoder on and off to save power.

Figure 4 shows the nWatch MP3 player application.

Bluetooth Low Energy (BLE)

The Bluetooth module (IC2) used in the nWatch is based on Texas Instruments' Bluetooth LE CC2540 chip, and can be used to send and receive data from a smartphone. This allows you to display notifications, or to update the time and date from the phone. I chose this module mostly because it was a great compromise between price and size. The range depends on many factors, and can also be changed by setting the registers responsible for the transmitting power. The range is actually quite good for a small PCB antenna. The module can wake up the micro when data is being received; data is transferred between the micro and the BLE module over a simple serial connection.

An inconvenience of this module is that it draws significantly more current when it remains connected. It is therefore better to disconnect after every data exchange and only reconnect when needed.

Sensors

The nWatch is equipped with three position sensors which I chose for being really popular among e-enthusiasts and so there are a lot of open-source libraries for them on the Internet. It's a "traditional" set of a gyro, accelerometer and magnetometer.

The first two sensors are combined in IC8, the MPU6050, a six-axis motion tracking device from InvenSense. It communicates with the CPU over an I²C bus, and has a separate LDO voltage regula-

tor to power it even when the main DC/DC converter is off. The interesting thing of this chip is that it has an on-board programmable digital motion processor (DMP). Consequently we can change its firmware, and let it do some calculations, freeing up some processing power of the main CPU. The firmware is uploaded to the sensor just before the initialization. Doing so gives the sensor more functions like tap and free-fall detection. Interrupts can be generated for such events and can be used to wake up the CPU.

The 3-axis magnetometer is the popular Honeywell HMC5883 (IC7), a small chip in a QFN package that also communicates over an I²C bus, but not the same as IC8. The reason for this is that I wanted to keep traces as short as possible to avoid placing unnecessary space-hungry and hard-to-solder vias. Since the CPU has two I²C ports, I simply wired the sensors to the I²C bus nearest to them.

Real-time clock

Imagine you want to set an alarm to remind you of some upcoming event. If this event is too far away in the future and the watch is left on, it will run out of power before reaching the event. This situation can be greatly improved by using a low-power real-time clock (RTC) with alarm functions. Now all you have to do is setup the RTC's alarm before switching the watch off. Once the RTC reaches the alarm date and time, it will switch on the voltage regulator IC6 (by means of transistor T2) and the watch will come alive. A good RTC that I found is the AB0805 from Abracon (IC9). It has an I²C interface and consumes around 50 nA, much less than the CPU in deep sleep mode would ever attain.

Power supply

The nWatch being a wearable wristwatch, it had to be as low power as possible. A step in the right direction was to allow the 3.3-volts supply to be switched off altogether. To make this happen I used a voltage regulator (IC6) with an enable pin (SHDN) that controls the output of the device. There are four ways of switching the watch on (again). The most obvious and easiest-to-implement technique is by pressing pushbutton S3. When the button is pressed a positive voltage is applied to the SHDN pin through D5 and the regulator turns on, powering on the rest of the circuit too. D2, D3 and D4 fulfill the same functions as D5 but for the

combined gyro and accelerometer (IC8), the RTC (IC9) and the BLE module (IC2) respectively.

IC6, an MCP1603 from Microchip, is a high efficiency 500-mA, 2.0-MHz synchronous buck regulator intended for battery-powered applications, exactly what was needed here. An inductor (L1) and a capacitor (C8) is all that is needed to make it fly.

IC4 and IC5, two LDO voltage regulators provide power to the parts that are always on (unless the battery dies): IC2, IC8 and IC9. IC5 also provides power to the CPU to prevent that it will lose its battery-backed memory registers when the 3.3-V rail is switched off. These registers are a special protected area of CPU memory where data is held as long as there is a high-enough voltage on supplied to its VBAT pin.

IC4 powers the BLE module so that it will not interfere too much with the rest of the circuit.

As mentioned before, LDO voltage regulator (IC13) provides 1.8 V for the core of the MP3 decoder (IC14).

R22 and R25 form a voltage divider allowing the CPU to measure the battery level. This divider can be switched off by T4 and T5 in order to save power.

Li Po battery with charger

The nWatch is powered from a 150 mAh Li Po battery. Although it is small, it allows the watch to sleep on it for more than a week. In case you actually use the watch for checking the time every once in a while, one battery charge will keep the watch running for at least five days. The battery is equipped with a PCM (protection circuit module), which protects it against short circuits and deep discharge. When the battery is fully discharged the PCM disconnects the battery from the rest of the circuit until a charger is connected. When this happens all memory is cleared, including the one of the RTC that holds the date and time. To make restoring data easy I wrote a simple app for my smartphone (see below).

Battery charging is controlled by a single-cell charger IC, the Microchip MCP73831 (IC3), which handles the charging process automatically. It only needs a resistor to set the charge current.

Connectors

The nWatch has three main connectors: a microSD card slot (K1) on the bottom,

a micro USB connector (K2) and a 3-pin pinheader (TP5, TP6 & TP7) used to program the microcontroller with an ST-Link programmer and/or debug the running program. This is the best and fastest way of programming the watch.

The micro USB connector is used to charge the Li Po battery and allows exchanging some data over USB. When connected to a computer (Windows, Linux or Mac OS) the nWatch is detected as a mass storage device (MSD), and the contents of the SD card are displayed. The USB port also provides a way of uploading programs to the watch thanks to its built-in bootloader. This is, however, much slower compared to using the external programmer. For USB programming the executable (HEX or BIN) file must first be converted into a special DFU (device firmware update) format. It is not possible to do any debugging over USB, so treat USB programming more like an additional feature instead of your preferred software development port.

The PCB

The shape of the PCB is adapted to the 3D-printed case and to the 3.5-mm jack connector. To make the watch as thin as possible I mounted the jack connector "in" the PCB instead of on it. Doing so saves a lot of space, allowing the watch case to be less thick.

Even though the PCB is routed with rather thin traces (≥ 6 mil) it is possible to make it at home using the standard photo-resist method. Doing this, however, is something for the do-**everything**-yourself (DEY) die-hard because soldering the 200 or so vias by hand is not for the fainthearted.

The microcontroller in its large, 144-pin LQFP takes up a lot of PCB real estate, making the board quite dense and I don't believe that I could have made it any smaller. At some point I thought about switching to a 4- or even 6-layer PCB with the main chips in BGA packages, but I did not pursue this idea as the nWatch was supposed to become a low-cost development board for hobbyists, not just an expensive prototype. Smaller parts would also make board assembly much harder, while using BGA packages would require special equipment. I therefore decided that a two-layer board had to do. The position of the components on the PCB is essential when you design a watch, because space is at a premium. Try to put all the components with simi-

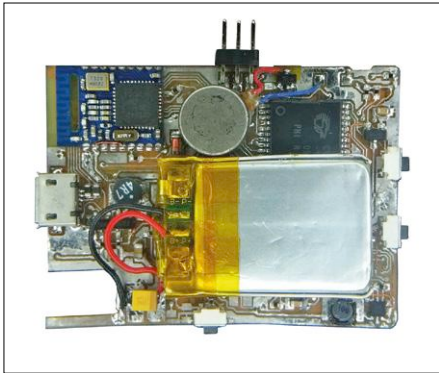


Figure 5. For a thin assembly the Li Po battery is placed on top of the SRAM chip.

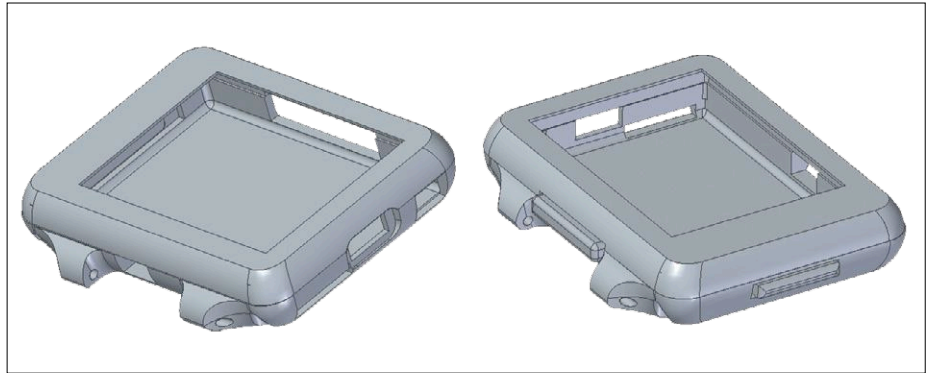


Figure 6. 3D renderings of the watch case.

lar heights on the same side of the PCB. If you mix them on bottom and top, the tallest part on each side determine the overall PCB height. A Li Po battery is also a problematic part. If you can, it is the best to put it in a milled slot in the PCB. When this is not possible due to the small board surface a possible solution is to make an area with only really flat components like resistors, etc. In my case I had no choice but to put the SRAM memory under the battery (**Figure 5**). It was the best compromise because it is only 1 mm thick. Other elements, like the 3.5 mm jack connector or the USB socket, may be “mid” mounted, meaning that they are hanging in a cut-out in the PCB. Doing so keeps the height of the board within bounds.

3D printed parts

3D printing is easy, the hard part is creating the 3D models of your design. Not only does it have to look nice, especially when you are designing a wristwatch, but the circuit board with all its parts has to fit in it too. Unfortunately, I have not discovered a magical solution to this problem, it still boils down to precisely measuring everything, 3D print a first

version and see where you went wrong when you try to assemble everything. You are probably going to need at least three or four iterations to get it right. Deciding to build a watch does not make things any easier as they tend to be rather small and thin. It starts by drawing a rough, overall shape for the watch, while keeping in mind the sizes of the PCB, the Li Po battery and the LCD with touch panel. Then you measure, try, curse and iterate until it all fits.

The parts that make up the nWatch case were designed in Solid Edge (**Figure 6**). Initially, the idea was to make the case at least splash-proof, but it turned out to be too hard to make all the connectors waterproof (especially the SD card slot). I had a few other ideas but eventually decided to make a two-part enclosure, held together by four tiny screws. This is a good solution, because after removing the screws we have an easy access to the main board, the Li Po battery, and the LCD with its touch panel.

The watch case has two rectangle slots for the buttons. The wake-up button is the biggest, so it is easy to press it and is placed above the watch band mount. The buttons are 3D printed as well.

The watch looks best when printed with a low layer height, meaning that the vertical step size should be really small. With a 0.2 mm z-axis step size I obtained satisfying results. My Prusa Mendel Iteration 2 (I2) 3D printer proved to have a good-enough resolution and stability to print a pretty nice case, showing that most 3D printers will be suitable for the job as the parts are rather easy to print.

Building your own nWatch

1. Get all the parts

Start by collecting the components, and preferably with the hard-to-get components. That way you quickly know where you stand. Apart from the touch panel, which is probably the trickiest part of the nWatch, there are a few other components that may give you a hard time finding them. The “mid-mount” 3.5-mm audio jack connector is such a part. I recovered it from an old tablet or smartphone of which the exact model and reference elude me, used it in my prototype, and then wasn’t able to find a drop-in replacement on the Internet. Those you can find either have a slightly different footprint, an incompatible plastic cover or both. However, it is possible to adjust

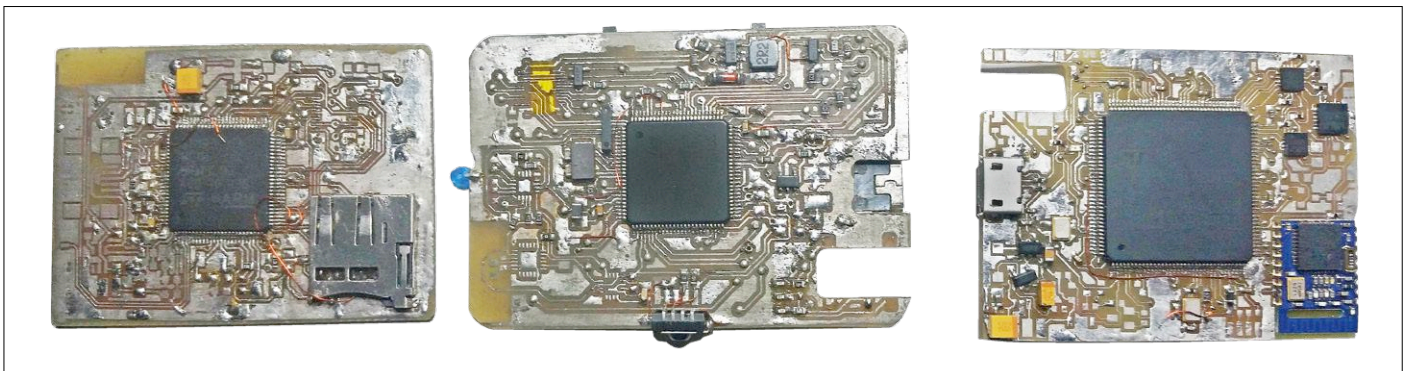


Figure 7. Several board iterations were needed before arriving at the final design of the nWatch.

the cut-out in the PCB with a tiny file, or connect the pads with bits of wire. The pushbuttons are less problematic, but may also be hard to find. I don't have an exact reference for them, just the dimensions. The best way to proceed is by comparing the dimensions and the way they look.

2. Assemble the PCB

Once all the parts collected and the naked PCB ready on your desk, you can start the board assembly (**Figure 7**). Soldering most parts on the PCB shouldn't be very hard as long as you use a small soldering iron. The tiniest parts are the 0402 resistors and capacitors, which need a bit of practice to solder. A hot-air station is more or less required for mounting the SMD quartz crystals and chips in QFN packages with their pads on the bottom of the package. There is nothing to be afraid of when you first deal with QFNs. If you prepare things well, all you have to do is to position the chip roughly and apply heat to it; it will align all by itself. Remember not to keep the heat on for too long to prevent the part from burning. The rest is nothing more than normal SMD soldering. Make sure to check that all the pins of the larger ICs (micro, VS1053, etc.) are soldered correctly. Sometimes they look soldered, but in reality they hang a little above the PCB surface without touching the pad.

3. Put it all together

After assembling the PCB you should check if there aren't any short circuits. Once you are sure about your soldering work, connect the battery and stick it to the PCB on top of IC10, the SRAM chip. My touch panel came with adhesive tape on the panel edges, so I just had to remove the protecting film and stick it on the LCD. What may cause some problems, however, is the connection to the PCB. Different pin-outs on different touch panels may require guessing the order of the four pins. In my case I had to cross two wires to make it work properly. Luckily, wiring it up the wrong way doesn't do any harm, so take as many guesses as you needed to get it right. If the coordinates are reversed, or touches are not detected, simply swap wires until it works as expected (**Figure 8**).

After soldering it to the main board, just put the LCD connector in place and put the display on the other side of the PCB, covering the battery.

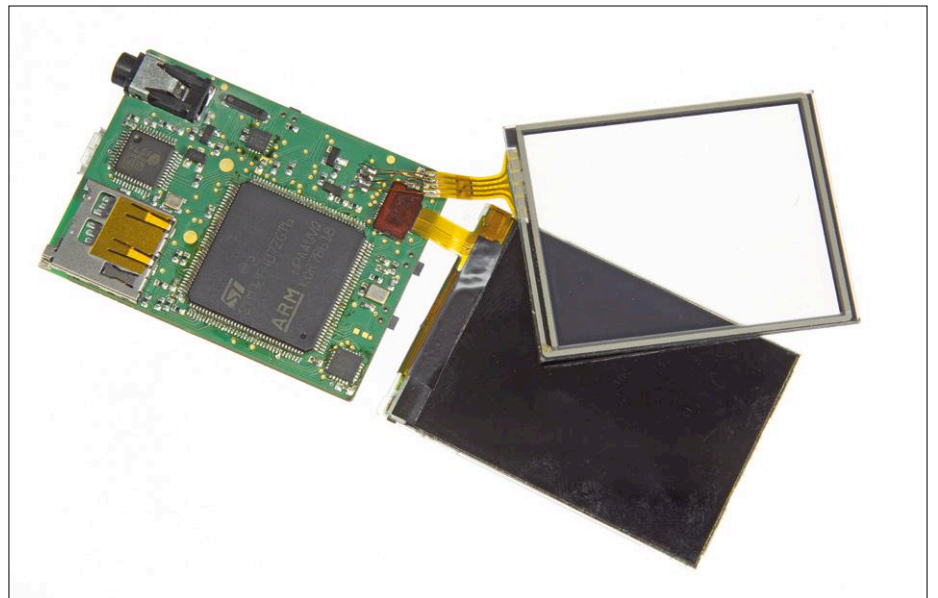


Figure 8. Assembled prototype with display and touch screen attached.

Case assembly is really simple (**Figure 9**). With a piece of sandpaper grind the surfaces where the two main body parts will meet. This is essential for obtaining a perfect fit. Also it is a good idea to drill out the holes in the bottom part and in the band holder, because after printing they are usually a bit obstructed with printing filament. Then insert the side button followed by the PCB (remember to remove the microSD card first). Place the wake-up button in its slot, and cover everything with the top part of the case. Insert and fix the four screws, and there you are.

For the watch strap or bracelet it should be possible to use any standard-sized

strap you like, but don't forget that you also need two of those watch strap spring bars.

Software

Software is a very essential part of every development board, and, of course, of a smart watch also. While I was developing and building different versions of the hardware, I also wrote some code to control the watch and adapted many libraries to work on the watch.

User interface with STemWin

One of the most important libraries for the nWatch is STemWin. This is a graphics library that controls the LCD,

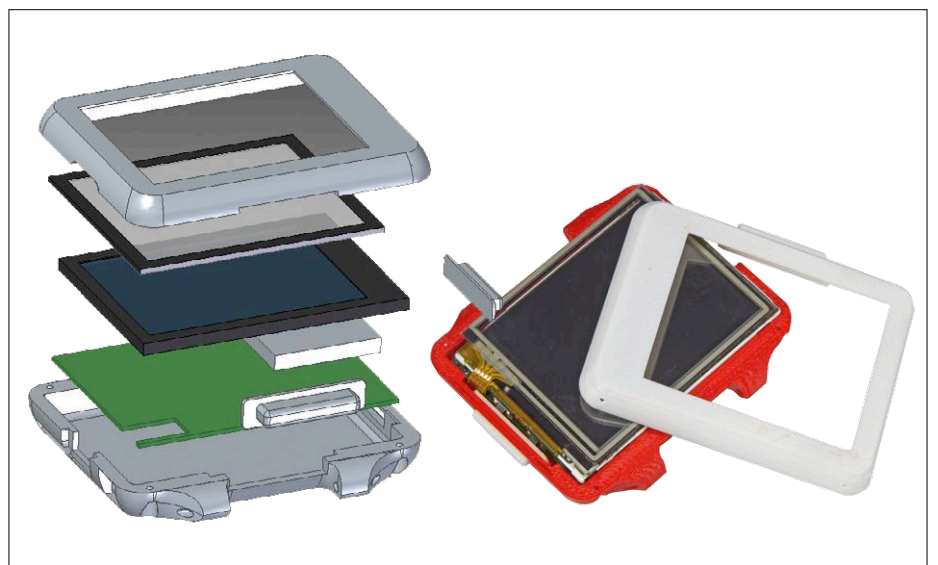


Figure 9. Stack the parts like this for the final assembly.



COMPONENT LIST

Resistors

R1,R3,R6,R9 = 0Ω, 0402
 R2,R5 = 10Ω, 0603
 R4,R7,R8,R14,R16-R21,R27-
 R30,R35,R36,R42,R46-R48,R55 = 1kΩ,
 0402
 R10-R13,R31,R32,R63,R66 = 4.7kΩ, 0402
 R15,R49-R51,R85 = 1MΩ, 0402
 R54 = 5.6kΩ, 0402
 R67 = 3.3kΩ, 0402

Capacitors

C1, C6 = 1μF, 0402
 C2,C7,C16,C18-C21,C39 = 100nF, 0402
 C3 = 10μF, 0805
 C8,C9,C17,C25 = 1μF, 0805
 C10,C11,C40,C41 = 22pF, 0402
 C26 = 4.7μF, 0603
 C29 = 2.2nF, 0603
 C35 = 47nF, 0402
 C36,C38 = 10nF, 0402

Inductors

L1 = 4.7μH, Murata LQM31PN4R7M00L, 1206

L2 = 22μH, Murata LQH31CN220K03L, 1206
 L3-L5 = 10μH, 0402

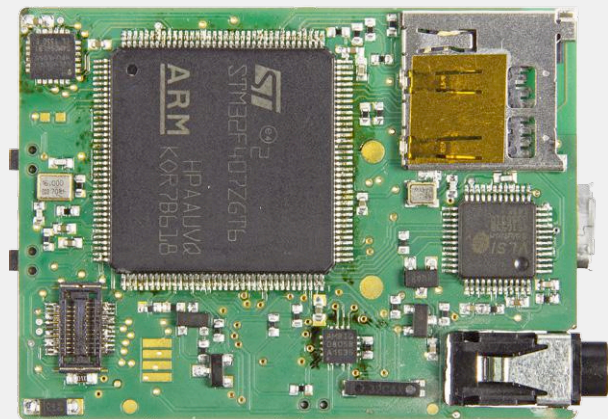
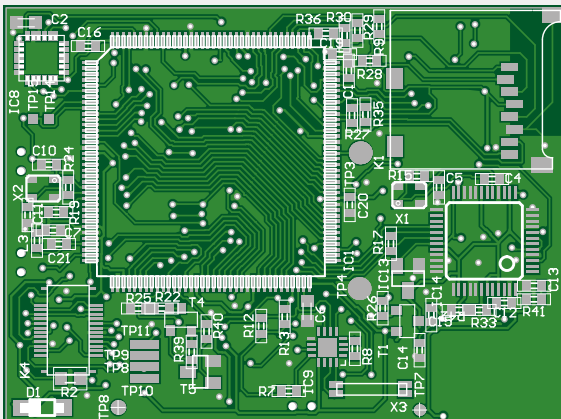
Semiconductors

D1-D5 = SBR1A40S1
 IC1 = STM32F407ZGT6
 IC2 = RF-BM-502
 IC3 = MCP73831
 IC4, IC5, IC15 = MCP1703T-3302E/CB
 IC6 = MCP1603T-3301/OS
 IC7 = HMC5883L
 IC8 = MPU-6050
 IC9 = AB0805
 IC10 = CY7C1041DV33-10ZSXI
 IC11 = STMPE811
 IC12 = AP5724
 IC14 = VS1053
 LED1 = LED, 0603
 T1-T4 = BSS84
 T5 = BSS123

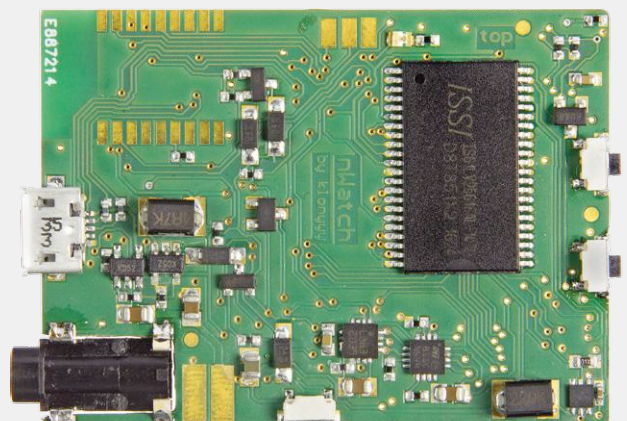
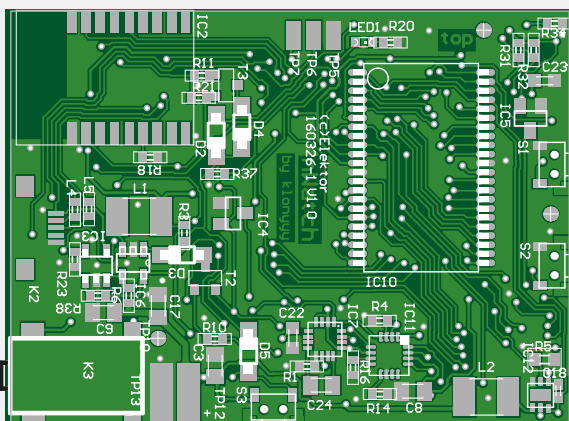
Miscellaneous

K1 = microSD card connector
 MEM2061-01-188-00-A.

K2 = micro USB Type B connector
 K3 = 3.5mm stereo jack connector
 K4 = Nokia 6300 LCD (or equivalent) +
 connector
 S1-S3 = tactile switch, side-mount,
 3.45×2.2×4mm, TACTDR345H40B160
 X1 = 12.288MHz crystal oscillator, SG-210STF
 X2 = 16MHz quartz crystal, 3.2×2.5mm
 X5 = 32.768kHz quartz crystal, MC-146
 2" resistive touch pad
 3D printed case
 Watch band + 2 spring bars
 150 mAh Li Po battery
 PCB #160326-1



PCB pictured at 150% of actual size



and draws all the elements of the user interface. Based on Segger's emWin, it is supported by STM (the manufacturer of the MCU) making it is easy to adapt to any STM32-based device. On the Internet you can find a lot of different examples and full documentation.

Looking at the sample programs, the screen elements such as buttons and text boxes will remind you of the old Windows look. However, and fortunately, these parts of the interface are fully customizable, the reason why this library is so interesting. Together with the library you get several utilities that help you to create the interface look and feel you want. GUIBuilder is, in my opinion, the most useful tool of all. It allows you to design a screen, and generate the code for it to add to your project. For sure, this intuitive tool is a real time saver.

Multitasking with FreeRTOS

This free, real-time operating system (RTOS), well-known to hobbyists and professionals alike, controls the watch. Thanks to its multitasking capabilities several applications can run simultaneously. It allows you to set task priorities, put tasks to sleep, give them a certain time to work and much more. I will not go into detail here because it is really well-documented software, please consult the Internet if you want to find out more about it.

Play animations with libjpeg

Another classic library I used is libjpeg, a library to decode JPEG files. I used it for playing simple movies on the watch. Its main advantage is its speed; it allowed me to achieve up to 21 frames per second (FPS) with 240x320-pixel frames. However, it also has an important drawback: it needs a lot of stack memory as it was originally a library developed to promote the JPEG standard. In cases where you need really fast JPEG decoding and you do not have to care about memory usage, this is the right library to use. In situations where memory is an issue it is better to use for instance the TJpegDec library.

Low memory-footprint graphics library

In situations where the libjpeg library is not suitable, you can use the Tiny JPEG Decompressor library TJpegDec. This tiny library, optimized for small, embedded systems was created by ChaN, a Japa-

nese developer well-known for his popular FatFS and petit FatFS libraries that are so very useful when using SD cards in embedded systems. Since TJpegDec was designed for small microcontroller systems, it does not require as much memory as libjpeg. It is fast enough to render for instance a photo gallery, but playing a video is pushing things too far.

The hardware configuration & ST libraries

The STM low-level libraries, although very complete, do not have a good reputation. Some users feel that they are inefficient and think that bare-metal programming is the only way to go, others only use the standard peripheral library. A general consensus has never been reached. Personally I had no difficulties in using these libraries, although I sometimes had to configure a hardware register myself, for instance in the case of periodically called function that has to execute fast (like a DMA transfer). If you are using the latest hardware abstraction layer (HAL) library from STM, you will have to adapt my code to add all the needed header files, etc. Some people have started using the STM32CubeMX initialization code generator tool to cre-



ate HAL functions but I have no experience with it, so you are on your own here.

Integrated development environment

The integrated development environment (IDE) that I initially used for writing the nWatch software was CoIDE from CoCox. I chose this IDE because I wanted unlimited code size and an Eclipse-like interface. I have no complains about the IDE itself, but it seems to be a bit forgotten by its creators. The last version that was released is buggy and I was obliged to stick to the previous version. In the end I switched to Atmel's TrueSTUDIO which seems to be continuously improved and supported. There are some other programming environments that you can use, including a bare Eclipse with toolchain, but they sometimes have code size limitations or are only available

About the Author

Peter (Piotrek in his homeland) is a 19-year-old highschool student living in Poland. He is an embedded systems enthusiast and enjoys microcontroller programming, especially AVR and STM32 types. He started his electronics career with building sumo robots for competitions, and recently got interested in wearable technology. Time permitting he also likes to build RC planes.

Fast Forward Award 2016

The nWatch was one of the entries for the *Elektor Fast Forward Award* organized by Elektor in collaboration with the governors of the 2016 *electronica* tradeshow in Munich. It was a great opportunity for hobbyists and professionals alike to share and present projects, products and startups.

The author of this article presented the nWatch project to an international professional audience and jury. This gave him the opportunity to meet several manufacturers to discuss the components used in the watch. Many opinions and suggestions from other participants and visitors made Peter leave as a happy man with a head full of new ideas. The fact that he won third place in the FFA's Prototyping category only made his smile bigger.

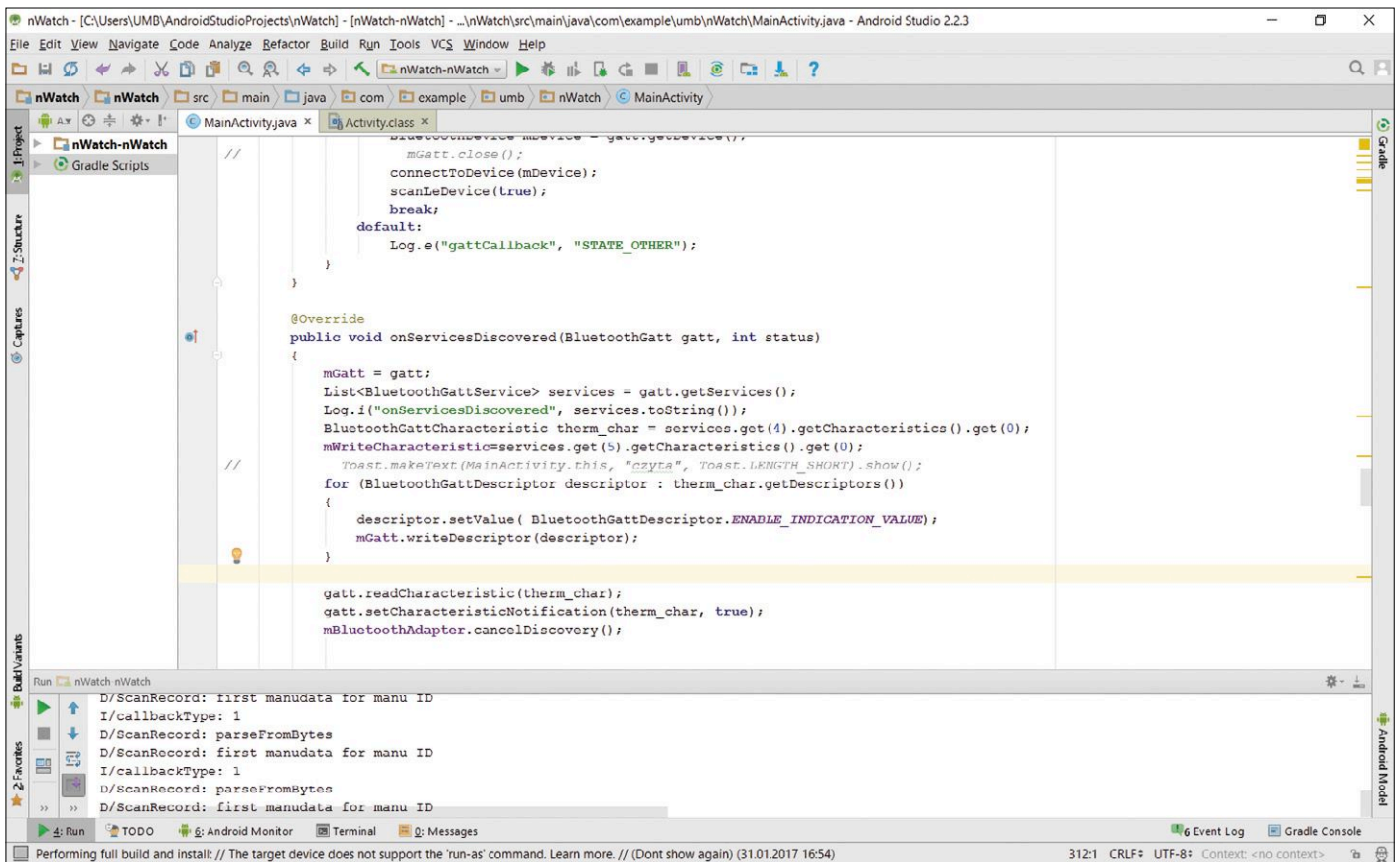


Figure 10. Android Studio 2.2.3 was used to create a simple Android app to communicate with the watch.

as trial versions. Configuring Eclipse for the nWatch all by yourself might cause headaches and may discourage you.

In-circuit programming

I highly recommend investing in a cheap ST-Link programmer. You will see that not only is it much more comfortable to use compared to the USB bootloader, it also allows you to debug the code you wrote. Being part of STM Discovery and Nucleo boards (that are often handed out for free at trade shows), make it the best candidate. All you have to do is make a

cable with a 3-pin, 2 mm female connector and you are ready to go.

Android app

Even though I'm not much of a Java programmer, I decided to write a simple app to send data to the watch from a smartphone. I found an example of a Bluetooth low energy app and modified it so that I could read the time and date from the phone and wirelessly send it to the watch (**Figure 10** & **Figure 11**). It is really comfortable, because there is no need to write a special app for the watch. When you launch the app on the smartphone it will look for an nWatch nearby, and when it finds one just tap the send button to set the time and date. The next version of the app should update the watch without any user intervention.

Conclusion

I think I managed to achieve my initial objectives. The device I built was a great learning experience that connected of many skills like programming, electronics, mechanics, and 3D design and printing. It was made of low-cost parts, some of them scavenged from old cellphones (unfortunately making reproducing the

watch rather complicated). Solving all the problems encountered during development of the watch taught me many things. Now I appreciate how much time and effort is needed to build a device like this. I also learned that teamwork is a really good thing as I wasn't able to make every part of this project perfect myself. However, I think the outcome is quite interesting.

With this article I wanted to show that everyone can build a device like this watch. It is not that hard after all and, in my opinion, it is the easiest way of learning something new. I hope you enjoyed reading the article as much as I did building the nWatch. ◀

(160326)

Web Link

[1] www.elektormagazine.com/160326

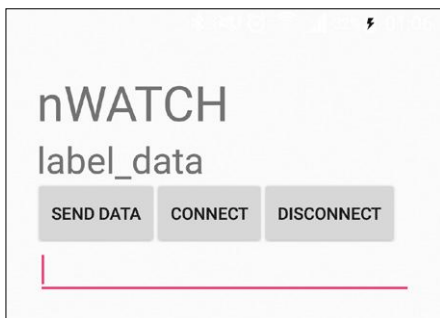


Figure 11. The Android app does not (yet) have fancy graphics, but it allows you to set the nWatch time and date by a tap on the 'Send Data' button.

FROM THE STORE

→ 160326-1
Unpopulated nWatch PCB



HomeLab Helicopter

Compiled by **Clemens Valens** (Elektor Labs)

Arduino Create

Although it has not stirred up a lot of dust yet, the popular Arduino platform is evolving in a rather spectacular way. Over the past months the people at Arduino have been quietly running a background task with the objective to develop a full-fledged cloud-based online development platform called *Arduino Create*. It is an ambitious project, taking the concept of the now defunct Codebender a step further, up to the prefab BBC micro:bit level. Regular visitors of the Arduino.cc website have had access to this platform almost from the beginning, but its presence was not advertised too much.

Arduino Create currently has two

knows only one board). The difference with those platforms is that Arduino Create can, like Codebender could, program the board itself without the user having to manually copy a file to a disk. To make this work, a piece of software giving the browser access to the board has to be installed first, so it is not completely hassle-free, but once that part properly in place the rest is automatic.

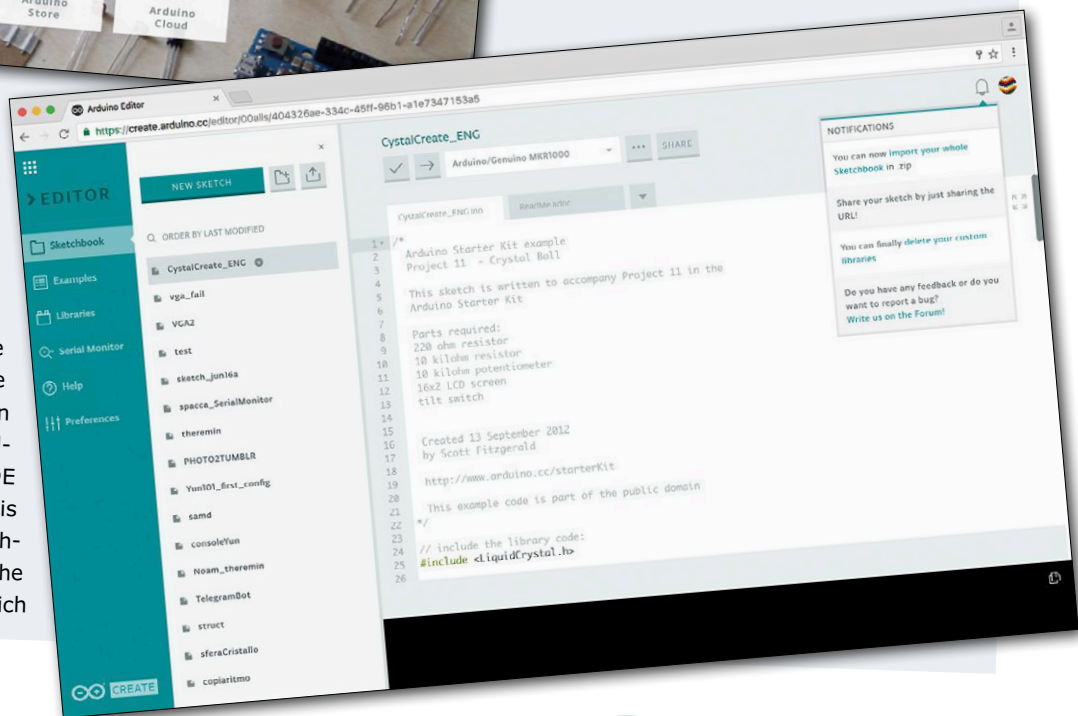
The second objective is to make the development of IoT applications as easy as possible by providing low-threshold cloud services for free. Again, this is very much like mbed, but accessible to non-specialists.

Traditional Arduino aficionados can continue using the off-line IDE they got to know so well. However, when downloading a new version from the website they will notice that Arduino Create is being 'pushed' as the preferred way to go. We look forward to what the future will bring. Arduino OS maybe?

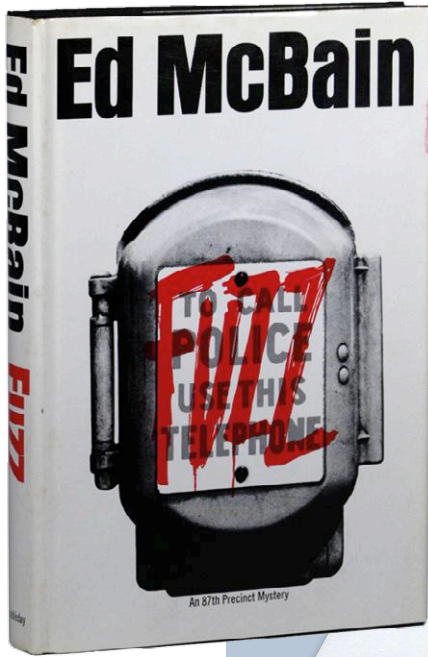
(160404-a)



main goals. The first is to completely remove the hassle of installing boards, tools and libraries. The Arduino Create user logs into his/her account, connects a compatible board to the computer and starts coding — it should be that simple. The board is recognized by the online IDE, all the libraries published in the Arduino repository are available without installing, the IDE is always up to date, etc. This is very much like ARM's mbed (without the programmer's jargon) or the BBC micro:bit environment (which



8,400,000,000
8.4 billion connected objects will be in use in 2017

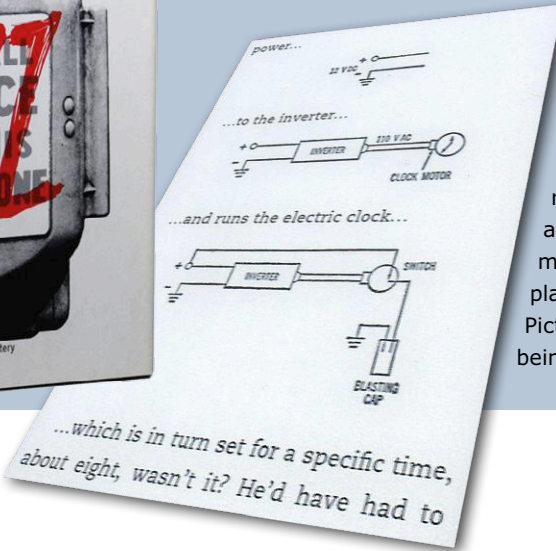


Electronics in literature

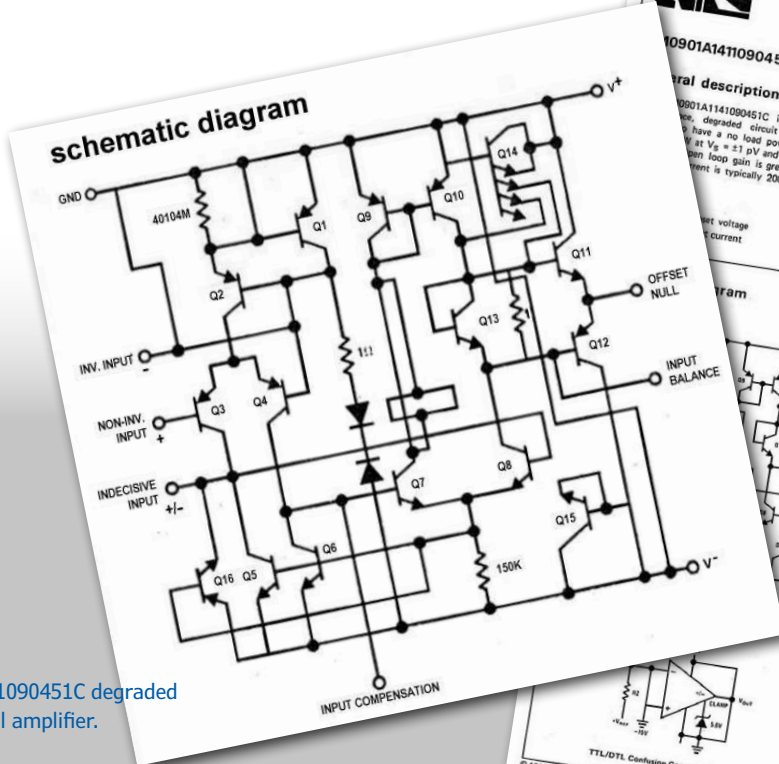
As far as I know — but what do I know? — literature and electronics do not mingle well. There is, of course, the Darlington substitution scandal situated in 1886, and involving Sherlock Holmes, but it was written many years before someone came up with the idea of the Darlington transistor (this feat from 1953 is, by the way, attributed to Sydney Darlington), and is therefore also much older than the admittedly bad idea of substituting one incorrectly, and although many stories feature electronics and technology in spectacular ways, they merely tend to show off the ignorance of the writers on the subject. Imagine then my surprise when recently reading a crime novel (Fuzz by Ed McBain, an 87th Precinct novel from 1968) I suddenly found myself gazing at circuit diagrams and a detailed and plausible explanation of how it was supposed to work. Pictures in novels are rare, what are the chances of it being a schematic?

(160404-b)

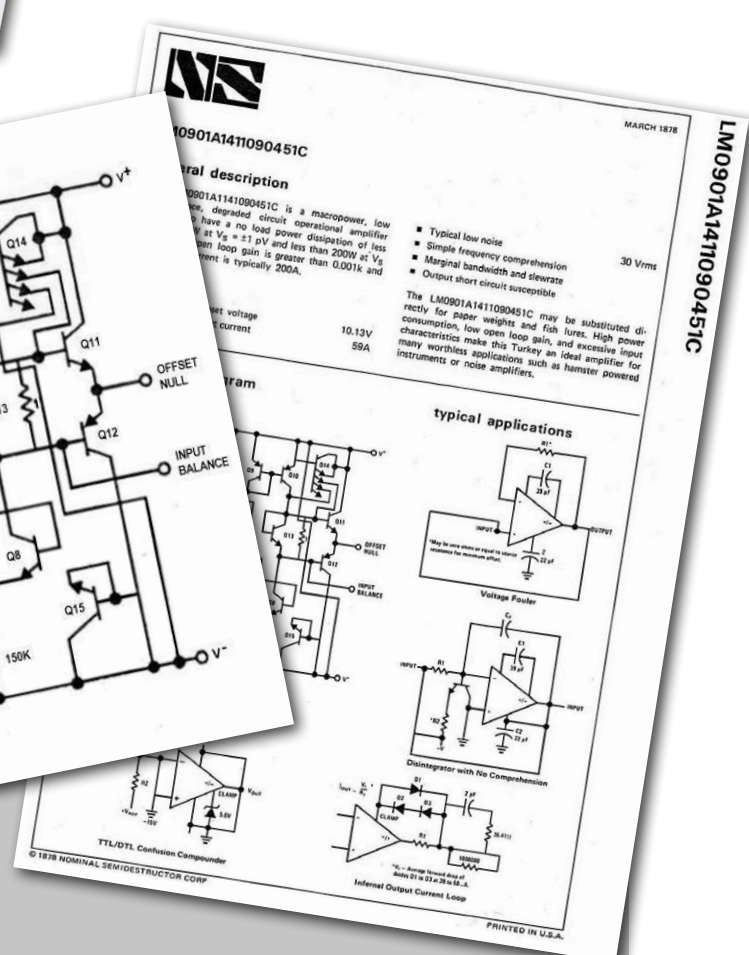
Fuzz by Ed McBain (1968)



...which is in turn set for a specific time, about eight, wasn't it? He'd have had to



The LM0901A1411090451C degraded circuit operational amplifier.



Weird components

The LM0901A1411090451C Polish operational amplifier is a macro-power, low performance, degraded circuit operational amplifier designed to have a no-load power dissipation of less than 0.553 W at $V_s = \pm 1$ pV and less than 200 W at $V_s = \pm 2$ pV. Open loop gain is greater than 0.001k and input bias current is typically 200 A.

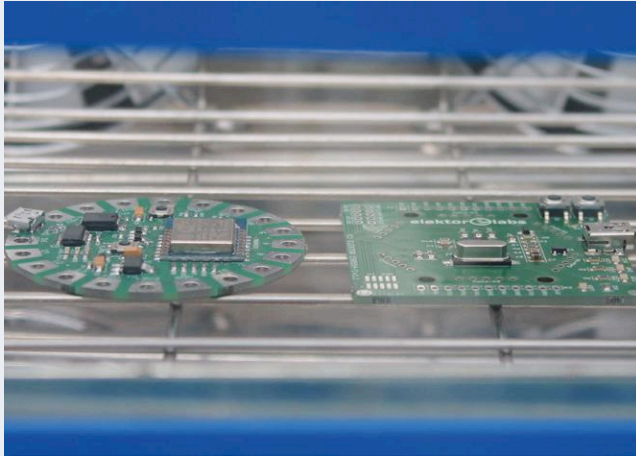
© March 1878, Nominal Semidestructor Corporation

<https://goo.gl/jfF75N>

(160404-c)

Improve your reflow skills

You may not have noticed it, but the main theme of this edition is printed circuit board (PCB) design. Since PCBs are closely related to soldering, the art of soldering is an important skill to master, even for a PCB designer as the technique used for soldering has an impact on the PCB design.



When surface-mount devices are soldered in a reflow oven, the temperature has to follow a precise path — the thermal or temperature profile — to ensure proper solder joints. Temperature profiles for components can be found in their datasheets, and not all components require the same profile. However, all parts, irrespective of their size and mass, must, within a defined time period, achieve the minimum temperature to start the reflow process without exceeding the maximum temperature the part (and others) can handle. It is easy to see that this is a complicated process. A standard for thermal profiling called IPC-7530 has existed since 2001. Sadly it is now obsolete; revision A is supposed to see the light soon. Even though its name suggests an update, it is actually more of a complete rewrite. The new document provides a myriad of information, from definitions to comparisons of solder, from techniques for measuring thermal profiles or for choosing a reflow oven to understanding solder reflow defects. The purpose of this standard is to provide useful and practical information to those responsible for developing thermal profiles to produce acceptable tin-lead and lead-free electronics assemblies. Officially the standard is targeted at managers, design and process engineers, and technicians who deal with mass soldering processes, but the homelab worker and other occasional reflow soldiers can use this document to greatly improve his/her reflowing skills. People who open the oven during the cooling phase to make a board cool down much faster may want to read this document to find out why they should exert more patience.

At the time of writing this item, the August 2016 draft of the document was available for free downloading from <https://goo.gl/I88pjX>

(160404-e)

Must-have homelab tool

If you have ever found yourself in a situation where you needed a dentist's mirror-on-a-stick to troubleshoot some inaccessible device, these glasses may be exactly what you need. With its prisms instead of normal lenses they allow you to look down while looking forward, hence see things in spots where your head can't go. Intended for reading on the beach without having to lift your weighty head, or for discretely spying on other people, prism glasses are great fun. How about rotating one of the prisms 180°? Now you can look up and down at the same time! Or rotate one 90° to the left and the other to the right and experience horse vision. Available all over the Internet, search for Prism Glasses or Bed Prism Spectacles.



(160404-d)



Source: Jannis Hermanns, <https://jann.is/>

Brick your computer

When you don't have access to CNC machines and laser cutters, and if you lack good drilling & cutting skills you can always turn to Lego to build an enclosure for your circuit. Many people do this for their Arduino or Raspberry Pi for instance, and I have even seen etching tools made with the little colorful bricks. Jannis Hermanns from Berlin (Germany) built a tiny Macintosh Classic "clone" out of Lego, a Raspberry Pi and an e-paper display. He used Lego Digital Designer (LDD) to design his bricked enclosure.

LDD can be downloaded for free from

<http://ldd.lego.com>

(160404-f)



Want to participate? Please send your comments, suggestions, tips and tricks to labs@elektor.com

Get 5 V from One Exhausted Alkaline Cell

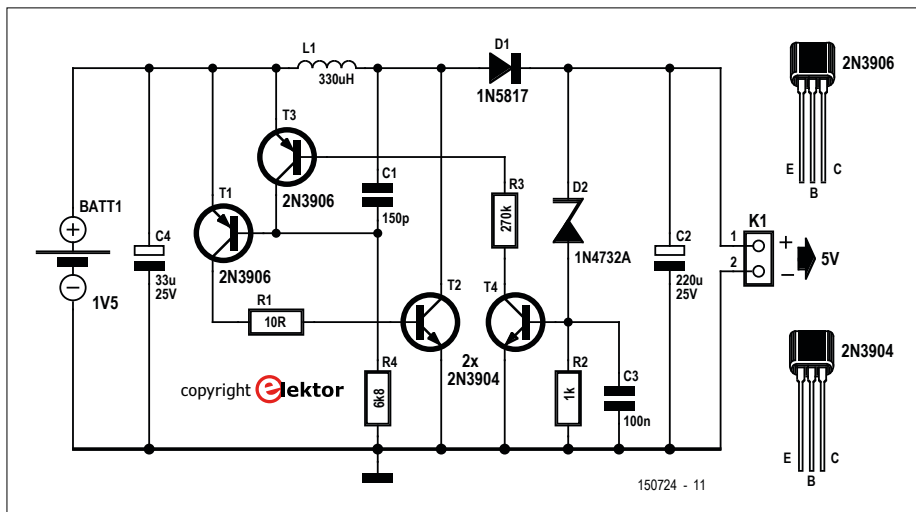
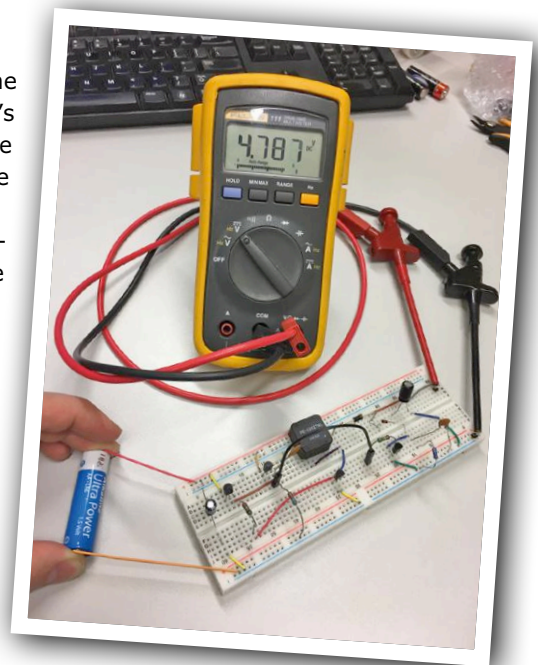
Juice it to the max

Every day we throw away tons of supposedly empty 1.5-volt alkaline (“dry”) cells. Triggered by the waste caused by my battery-powered equipment failing to work with partially discharged cells, I designed a circuit to step up the voltage of one exhausted cell to around 5 V for use in another application. Say, to make an LED light, keep a lightweight microcontroller buzzing, or power remote sensors that do not demand much energy... Until that battery, like The Parrot, is **dead**.

By **Juan Canton** (México)

This self-oscillating voltage step-up converter supplies an output voltage just under 5 volts from a battery voltage down to about 0.8 volts. At about 60% efficiency and just 5 mA of output current it’s is not a wonder of efficiency but that was not the intention. Really, the author was after juicing exhausted 1.5-V batteries to the maximum using components readily available to him at low or even no cost, as part of an experiment. The critical factors in the circuit are inductor L1 and the transistors. On the latter, you might think the 2N3906 and 2N3904 are the TUP and TUN of the Americas and can be substituted by

BC560 and BC550-ish devices at the drop of a hat. No cigar, at least that’s what Elektor Labs users reported. The stray capacitance and other device parameters will play tricks on you. In essence you are looking at an oscillator T1-T2-L1-C1 controlled to some extent by T4 in terms of the output voltage appearing on K1. Coil (not: choke) L1 is the energy storage device, helped by C2 as a reservoir and smoothing device. Of the “interesting” parts, the 1N4732A is a ‘voltage regulator diode’ specified by NXP for nominally 4.7 V, 53 mA I_{fwd} and 10 μ A I_{rev} at 1 V V_{rev} . L1 then is a true inductor, not a choke. Our labs trainee’s selection of the part

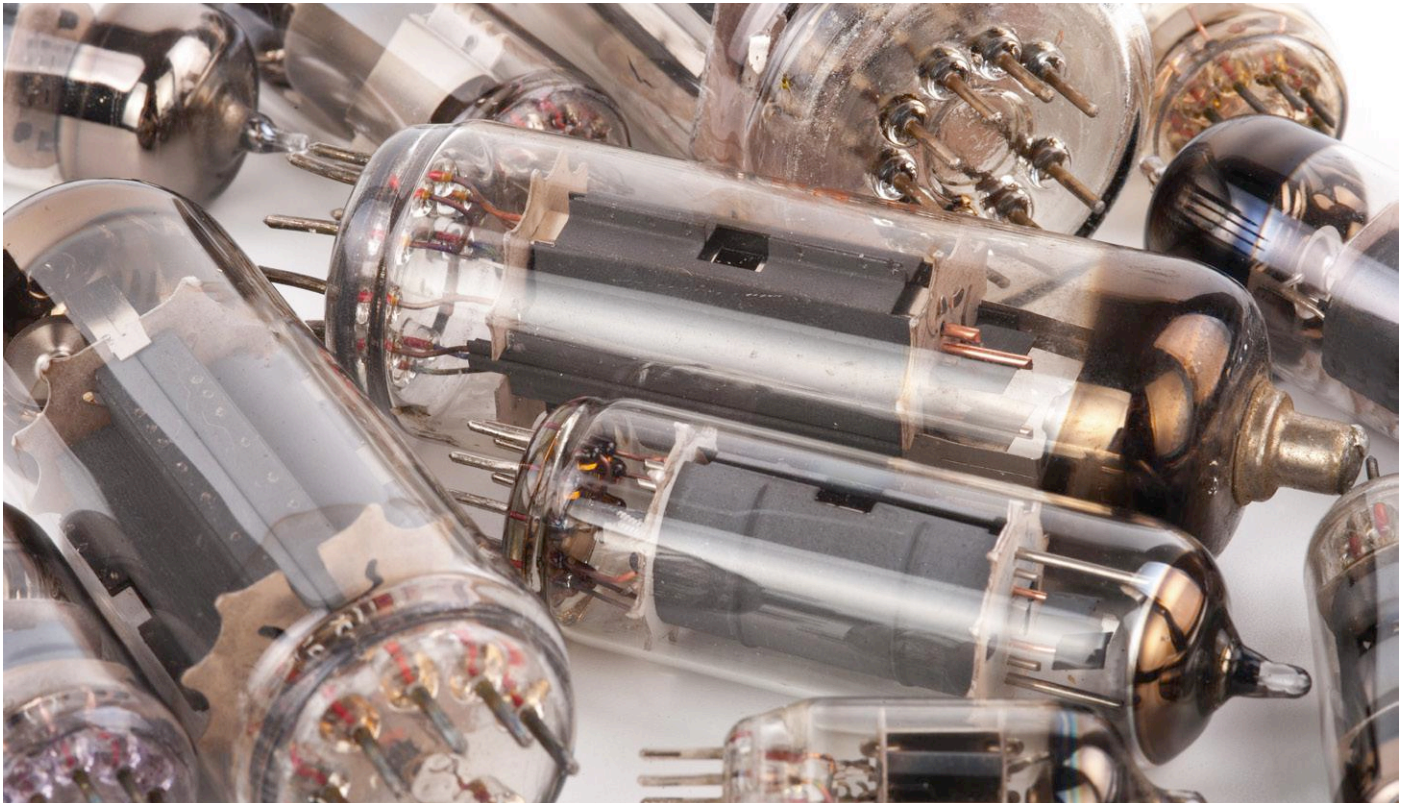


worked satisfactorily with a promise of better results by using a slightly higher value. Do experiment and give it a try. The PE52627NL from Pulse Engineering (#1209550 at Farnell) is specified for 0.78 ohms max. DC resistance. It’s a low-profile and compact device. The 1N5817 is a 1-amp Schottky (i.e. low-dropout) rectifier diode. Do not use a regular silicon 1N400x in this position as it will waste output voltage. Interestingly, you can power the circuit with a higher voltage without exceeding the output voltage and the load coming to grief — the network that regulates the output voltage will stop the oscillation. All within limits of course. ◀

Figure 1. An experimental 5-V step-up converter for exhausted 1.5-volt batteries.

(150724)

Vacuum Tube Curve Tracer 'Remake' of the Tektronix 570



In order to test salvaged vacuum tubes, the author required a tube tester. Because a 'real' tester like the Tek 570 was an unattainable ideal, he decided to build one himself — based on a modern microcontroller. The version presented here can measure the cathode current as a function of either anode voltage or grid voltage.

By **Charles van den Ouweland**
(Netherlands)

Do you suffer from this too? That you can't actually throw anything away, and therefore from of all those hopelessly defective 'Retronics' devices, donations from family, friends or neighbors (three categories that do not necessarily overlap) who thought that they were doing you a favor, you occasionally try to salvage some useful part? The author too, and after a number of years there was

a large collection of vacuum tubes in the attic, which looked visually intact and that were really only collecting dust. What do you do with a chest full of parts salvaged from old equipment? You use them, of course – for quick & dirty experiments they are often perfectly usable. Things like resistors, capacitors, inductors, etcetera are measured easily enough, and also for transistors

Figure 1. This is what the characteristics produced by the vacuum tube tester a.k.a. "tracer" look like.

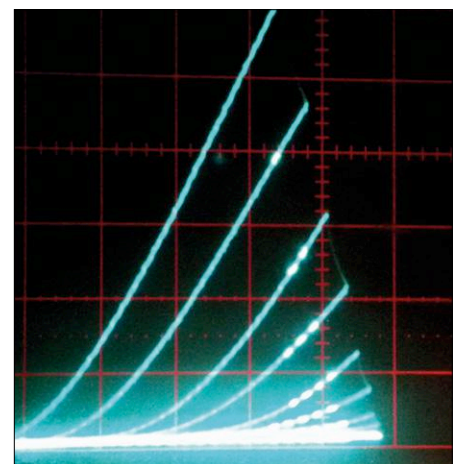




Figure 2. The original Tektronix 570 is a monster in the truest sense of the word.

and diodes their usefulness can be ascertained quickly. With vacuum tubes this is not that simple, however. Okay, severe damage may be recognizable with the

naked eye, and a potential short circuit between electrodes could be detected using an ohmmeter, but that is about all. On the other hand, there is little point in building a circuit and using a tube of which it is unknown whether it is "good" or not, because when the circuit then doesn't work, where is the problem likely to be? The design, the defective tube, or both?

The Tektronix 570

Even if you have no intention of doing something yourself with these tubes, you can always try to sell them. Browsing through eBay, the author came across a French vendor who accompanied every tube for sale with a photo of its characteristic, made with the aid of an old Tektronix 570 curve tracer.

To generate these characteristics (Figure 1) the tester applies different grid voltages to the tube under test (0 V, -1 V, -2 V, etcetera) and then varies the anode voltage from zero to a few

hundred volts. Then, on the screen of the cathode ray tube the cathode current is displayed as a function of anode voltage at different grid potentials. These characteristics make it clearly visible whether the tube is suitable for a certain application.

The first thing that came to the author's mind was "I want that too". The only problem is that the Tektronix 570 (Figure 2) has now become a collector's item; if there is even one to be found, then you will certainly need a well-filled wallet. Apart from that, the '570 is a real monster with dimensions of 42 x 33 x 62 centimeters and weighs around 80 pounds.

As with any right-minded electronics engineer this was immediately followed by a second thought "I can do this myself". With modern semiconductor and microcontroller technology and using an existing oscilloscope, this should be possible at a fraction of the cost (and also much lighter). The idea of the 'Neptronix

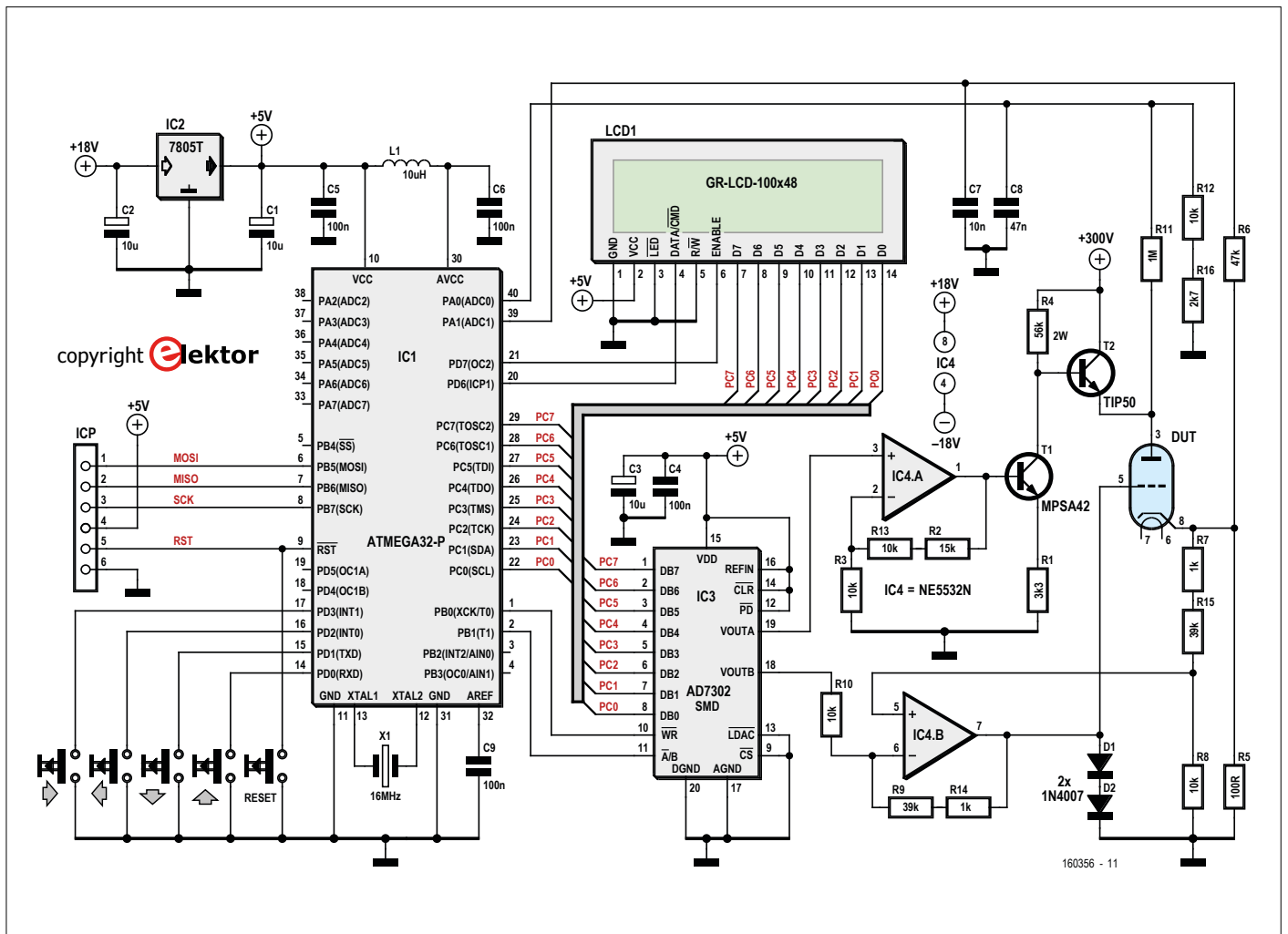


Figure 3. The schematic of the vacuum tube tester.

570' (where, 'nep' is Dutch for 'fake' or 'bogus') was born.

Design

The circuit has been designed such that two types of measurement are possible:

- varying the grid voltage with several different fixed values of anode voltage;
- varying the anode voltage with several different fixed values of grid voltage.

This is seamlessly controlled using a microcontroller and for the user interface serves a small LC Display with a set of four pushbuttons. The software has been designed in such a way that it is also possible to display the measured characteristics on the display instead of on the display of an external oscilloscope. The Neptronix 570 is and remains built on a breadboard; no circuit board has been designed for it.

Birds-eye view of the schematic

The schematic for the tracer is drawn in **Figure 3**. On the left is drawn the outline of IC1, the microcontroller — an ATmega32-P. This μ C was selected because it has a large number of I/O lines, is available in a DIP package for not much money, and because the author had a couple on hand.

To the left of IC1 we see the ICP programming connection and the four push-buttons for the user interface (and also the reset button). The 8-bit wide data bus from the μ C goes in one direction to the LC Display (LCD1) and in the other direction to a dual-channel D/A-converter (IC3). This provides the control voltages for the different drive voltages for the tube that is to be tested.

Finally, on the right, two opamps are drawn; the top one (IC4.A) supplies, via two high-voltage transistors, the anode voltage for the tube (DUT = *Device Under Test*), while the bottom one (IC4.B), generates the grid voltage.

Power supply

As is known, tubes generally operate at (very) high voltages. In the schematic several different power supply voltages are indicated – first let's take a look where these come from.

The power supply voltage for the digital electronics and the display (5 V) is supplied through a simple 7805T (IC2 in **Figure 3**), which in turn is powered from +18 V that comes from a lab power supply. Note: this 7805 needs to have

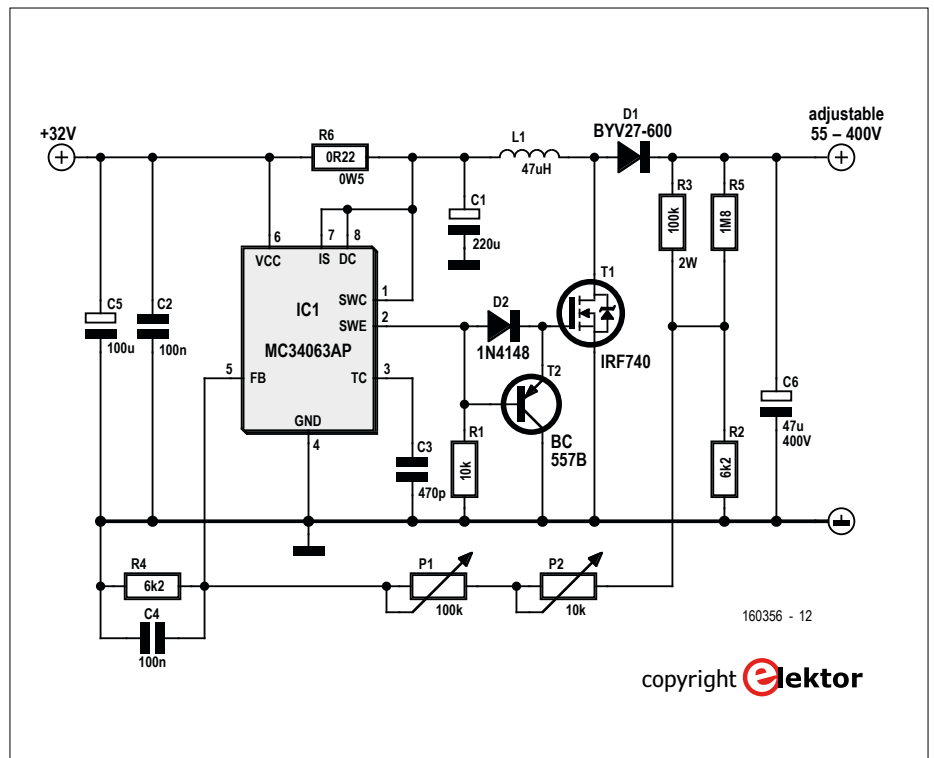


Figure 4. The high-voltage power supply; the input voltage of 32 V is supplied from an old printer power supply.

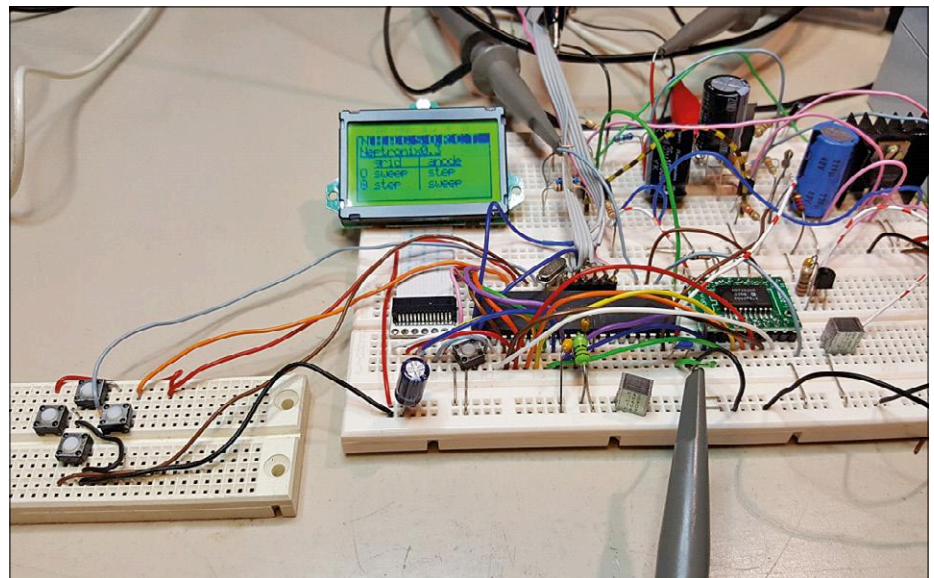


Figure 5. The vacuum tube tester as described in this article.

adequate cooling! The same lab power supply also generates the negative voltage of -18 V; this ± 18 V is used to power both opamps. Since everyone is likely to have a (dual) lab power supply on the workbench there is no need to go into any further details.

The high-voltage supply is a bit more complicated. Not everyone will have an adjustable DC high-voltage power supply

on hand (in any case, the author didn't), so a solution has to be found for that. A very big problem this fortunately is not – see the schematic of **Figure 4**. This is a standard application of the step-up converter MC34063AP.

The author used a power supply, salvaged from a worn-out Canon Pixma printer, that generates the input voltage of 32 V. However, if the available input



Figure 6. The Start tab.

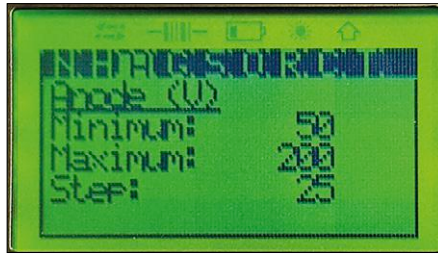


Figure 7. The Anode tab.



Figure 8. The Grid tab.



Figure 9. The Output tab.

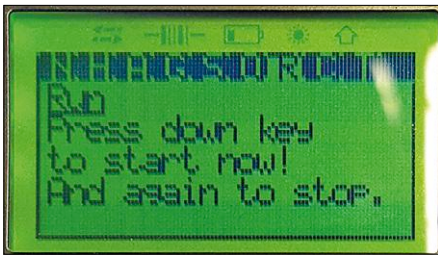


Figure 10. The Run tab.

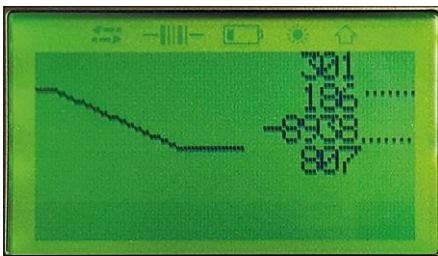


Figure 11. The calibration screen.



Figure 12. Calculation of the transconductance.

voltage is (much) lower, then it can be difficult to obtain a sufficiently high output voltage. In that case a voltage-doubler could offer a solution.

A few details

The (dual-channel) D/A-converter IC2 (an AD7302) is unfortunately not (at an acceptable price, at least) available in a DIP package. The author therefore mounted the SMD version of this IC on an adapter board, so that it would still be possible to build the tube tester on a breadboard without any difficulties.

The anode voltage of the tube to be tested amounts to maximum of around 320 V. For this purpose, opamp IC4.A multiplies the voltage at the output A of the DAC (0 to 5 V) by 3.5. Subsequently, T1 multiplies this voltage again with a ratio of 17. Transistor T2 ensures that sufficient current is also available.

When we only want to vary the anode voltage ('sweeping'), then it is not necessary to know the exact amount of gain from IC4.A and T1. There is, however, also a 'reversed' operating mode implemented (variation of the grid voltage for different fixed values of anode voltage) and for this the anode voltage needs to be exactly regulated to known voltages. For this reason, the voltage divider built from R11, R12 and R16 is used to feed back one-eightieth part of the anode voltage to A/D-input ADC0 of the microcontroller. In this way the controller can calibrate the anode voltage.

The other output of the DAC goes to opamp IC4.B. Because the grid voltage needs to be negative, this opamp converts the 0 to 5 V from output B of the DAC into a range of 0 to -18 V. Diodes D1 and D2 protect the circuit from voltages that are too high.

To measure the cathode current, a resistor R5 (100 Ω) is inserted between the cathode and ground. The voltage drop across R5 goes via R6 to the second A/D-input (ADC1) of the microcontrol-

ler. To compensate for the voltage drop across R5, voltage divider R7/R15/R8 applies one-fifth part of the cathode voltage to the non-inverting input of IC4.B. **Figure 5** gives an impression of the vacuum tube tracer/tester during the development phase.

User interface

The LC Display, together with four push-buttons (up, down, left, right), are used for operating the vacuum tube tester. The display shows a series of tabs in which different settings can be made. Navigating between the different tabs is accomplished with the left and right push-buttons; the corresponding tab is opened using the down pushbutton. Now the left and right buttons can be used to change the settings within that tab. Below is a brief description of the implemented tabs.

Start (N)

On the start screen (**Figure 6**) the selection can be made between the two operating modes:

- varying the grid voltage with different values of anode voltage (*step plate/anode, sweep grid*) or
- varying the anode voltage with different values of grid voltage (*step grid, sweep plate/anode*).

The right button advances to the next tab.

Anode (A)

In this tab (**Figure 7**) the minimum and maximum anode voltage and the step size of anode voltage are set. Up/down are used to navigate to a particular value, after which it can be changed using left/right. The step voltage is used when in the Start tab the selected mode is *step plate, sweep grid*. This value determines the amount that the minimum and maximum voltages are increased or decreased.

Grid (G)

Here the minimum and maximum grid voltages are set, as well as the grid step voltage (**Figure 8**). The latter is used when the operating mode *step grid, sweep plate/anode* has been selected. This value also determines the amount that the minimum and maximum are increased or decreased.

Output (O)

Here the selection is made for output to an oscilloscope or to the LC Display (**Figure 9**).

Run (R)

Push the down button on the tab (**Figure 10**) to start the measurement process. When the output is selected to go to the LC Display, the results will appear here. The sweep-parameter is displayed on the horizontal axis.

In this context a few remarks. If an oscilloscope is used to display the characteristics, then it has to be connected as follows. In the mode *sweep grid, step anode*: x-axis is grid, y-axis is cathode; in the mode *sweep anode, step grid*: x-axis is anode, y-axis is cathode.

In the LCD mode the measurements are performed relatively slowly, and the firmware calculates the average of a number of measurements before sending this to the display. On the other hand, in the oscilloscope mode the measurements are made quickly, to prevent flickering of the display.

Calibrate (C)

After a press on the down button this tab appears (**Figure 11**), showing a graph of the anode voltage (0 to 230 V) versus the DAC value (0 to 255). The numeric values in the example of **Figure 11** from top to bottom: maximum voltage at a DAC value of 0; minimum voltage at a DAC value of 255; gradient; intercept of the straight line with the y-axis.

After a press on the up button, this graph disappears and the theoretical curve appears (same axis).

Transconductance (T)

In this tab (**Figure 12**) the transconductance S can be measured. This works as follows: first the anode voltage is set to the maximum value (anode tab). The grid voltage is also set to the maximum value (grid tab). Now the current I_1 is measured. Subsequently, the grid voltage is lowered by one step value and the current I_2 is measured. The transconductance then follows from:

$$S = (I_1 - I_2) / \text{grid step voltage} \quad [\text{A} / \text{V}]$$

In the example of **Figure 12** the anode voltage is 200 V and the grid voltages are 0 V and -1 V. The measured values are $I_1 = 600 \mu\text{A}$ and $I_2 = 11000 \mu\text{A}$. This results in a transconductance of -10.4 mA/V.

The tabs H (filament voltage/current) and S (screen grid) were not yet implemented at the time of writing.

The firmware for the vacuum tube tester (which can be programmed into the microcontroller via the ICP interface using any regular AVR programmer) can ('as is') be downloaded from the Elektor Labs page for this project ('vacuum tube curve tracer'). There you will also be kept informed about future developments and updates, and you can add your own contributions and comments. ◀

(160356)

About the Author

Charles van den Ouweland (Breda, 1963) studied Electrical Engineering at the Eindhoven Technical University, Holland. He currently works as a software architect for ProRail, but in his spare time he keeps busy with electronics — in particular the fusion of modern microcontrollers with 'old fashioned' tubes.



Advertisement


Sales@ezpcb.com

EZ PCB

Welcome

www.ezpcb.com

One-Stop PCB & PCBA Turnkey Service



PCB Online Calculator
No Need to Register
Instant Quote & Pay



1 To 40 Layers
Prototype to Mass Production
Amateur to Professional



Prototype Start At \$5/PC
2L 4" x 4" each
Free Shipping

In the past years, our PCB have been shipped to 40 countries



Err-lectronics

Corrections, Updates and Feedback to published articles

BL600-eBoB (1)

Elektor 2/2015, p. 42 (140270)

UPDATE. The manufacturer of the Laird BL600 Bluetooth Low Energy Module which was featured in the original March & April 2015 article has since updated its software (firmware and libraries). Unfortunately the earlier version seems to be no longer available. The example given in the 2015 article was based on version 1.5.7.0 revision 5 of the firmware and its associated libraries.

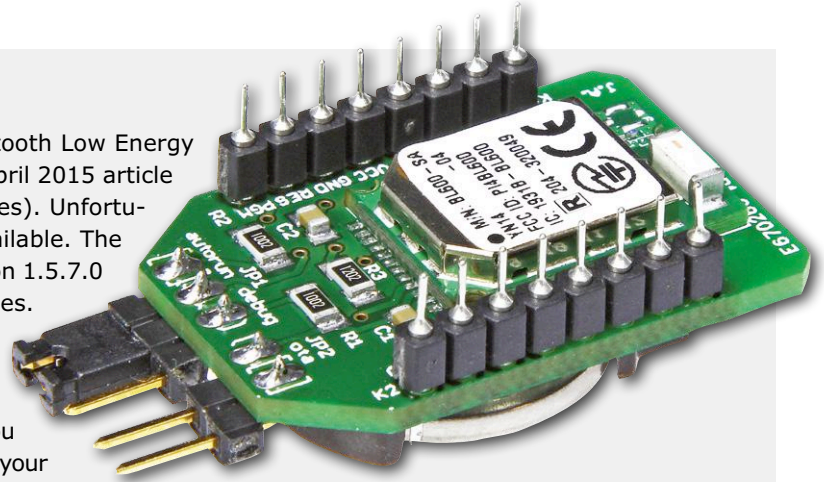
It could be that this example is not compatible with the latest version of the firmware and its libraries.

In any case, it is important to use only the library that matches the BL600 firmware and revision version. If you see a 'BL600: Cross Compiler not found' error message, your module firmware and downloaded library are not compatible.

The AT commands for identifying the firmware and revision of the BL600 (first connect the module via the serial interface to the PC) are:

- firmware: AT I 3<Enter>
- revision: AT I 0<Enter>

See the 'eBoB BL600 (2)' (www.elektormagazine.com/150014) article for further information on the firmware.



Welcome to the Share section

Elektor 1/2017, p. 114 (160252)

In this piece Elektor Netherlands editor Thijs Beckers wondered why the "polarity" of the AC line connection to a hot water boiler should be important. This generated lots of feedback for which we are grateful!

Below is a selection.

Readers should be aware that 230 VAC line outlets in Holland are earthed in wet rooms, or non-earthed in dry rooms. Neither type is polarized, i.e. the connection of the Live (L) and Neutral (N) pins in the outlet is not regulated. The Protective Earth (PE) pins are connected to a local earthing pin and to central earth at the power station and/or the substation.

FEEDBACK. The Live and Neutral connections are important when a single-pole switch is used to switch equipment on and off. One example I have noticed, with fluorescent tubes. The ballast is a sort of auto-transformer wired in series with the filaments at either end of the tube. When the on/off switch is in the Neutral wire the filament is at AC mains potential even when switched off and this voltage is sometimes sufficient to cause a low-level glow discharge in the gas around the filament to the earthed fitting. With the switch in the Live wire, the filament will be at AC neutral potential which is much closer to earth and no discharge or glow is visible.

Lothar Freißmann, Germany

FEEDBACK. The problem is probably caused by electrolysis. There is always some DC component on the (AC) power line and this causes deposits to form on electrodes which can be remedied to a certain extent by grounding the device. Reversing the plug reverses this effect; but deposits will then be formed on the other electrode in the heater. I have a similar problem here in Canada, namely a buildup of lime scale in my instantaneous water heater. Running a solution of vinegar through the pipe work for a couple of hours is effective in removing the lime scale. To make the

descaling process easier I have added a bypass and pump to the pipe work so that I can flush through with vinegar for as long as necessary.

Clinton Millet, Canada

FEEDBACK. Looking through the handbook for my old gas hot water boiler 'Calenta' made by the company Remeha 'mains phase reversal detection' is an option that can be enabled in the setup menu (parameter #43). In some countries the mains plug design allows it to be inserted in one of two ways only so that the Live and Neutral connection become swapped when the plug is inserted 'upside down'. Well naturally I had to try this menu option out. The default factory setting is with this phase detection disabled. With this option enabled I swapped the Live and Neutral wires and sure enough, on switch on it produced the error code SU9 'mains plug incorrectly wired'. In the boiler combustion chamber the igniter electrode produces a spark to start the gas burning and after ignition it is fed with a high voltage producing ionizing current which is measured to check the combustion conditions are within safe limits. The handbook states that this ionization current should be less than 3 μA . Replacement electrodes are included in many of the standard service kits for the boiler. The high voltage is produced by a step-up switching regulator which could possibly generate faults if the Live and Neutral wires are swapped. Measuring such low levels of current with a worn and dirty electrode will also eventually lead to unreliable operation.

Peter van de Meerendonk, The Netherlands



Opamp Power Supply Connections...

Elektor 1/2017, p. 118 (160257)

FEEDBACK. I work in IC design and development, and the problem of the IC variant marked with an 'R' indicating reversed supply rail pins is probably a result of an important customer of the chip requesting this specific pin assignment to simplify their own board layout. It's a simple job for the chip manufacturer to change the bonding pattern and chip labeling. As PCB component density increases alternative pinouts can be useful to solve a tricky layout problem.

Lothar Freißmann



Minuscule MEMS microphone — for the Bat DetectorPLUS

Elektor 6/2016, p. 115 (160083)

FEEDBACK. In the article it states that special ultrasonic microphones are very difficult to find. That's not strictly true :-). As an old 'night owl' I am also a bit of a bat fan (the creatures, not the superhero) and have built several bat signal detectors over the years. First purely digital (using a 7493 divider), later an analog down-converter design (using an SO42P). I needed microphones suitable for the application and found the ultrasonic module a good (and cheap) source. Try searching for 'HC-SR04' at an online auction site like eBay.

This module, otherwise known as a 'ping sensor' contains an ultrasonic microphone along with some other chips in SMD-outline that could be salvaged for use in other projects. It also has an ultrasonic transmitter or loudspeaker module which can be used for sending ultrasonic signals (as used with early TV remotes). These HC-SR04 modules are a popular choice for obstacle avoidance and distance sensing applications for robotics with Arduino systems. Some suppliers offer discounts on packs of 5. The transducers are bigger than the original unit and much easier to handle. They use standard electret transducers making them compatible with many different types of bat detector circuits.

Another advantage of their size is they can be easily fitted with a parabolic reflector (made from stiff paper) with the transducer positioned at the focal point. This makes the microphone directional and helps pinpoint the creature at a greater range. The microphones can detect signals beyond 100 kHz, but most native bats don't squawk at such high frequencies.

Peter Krengel



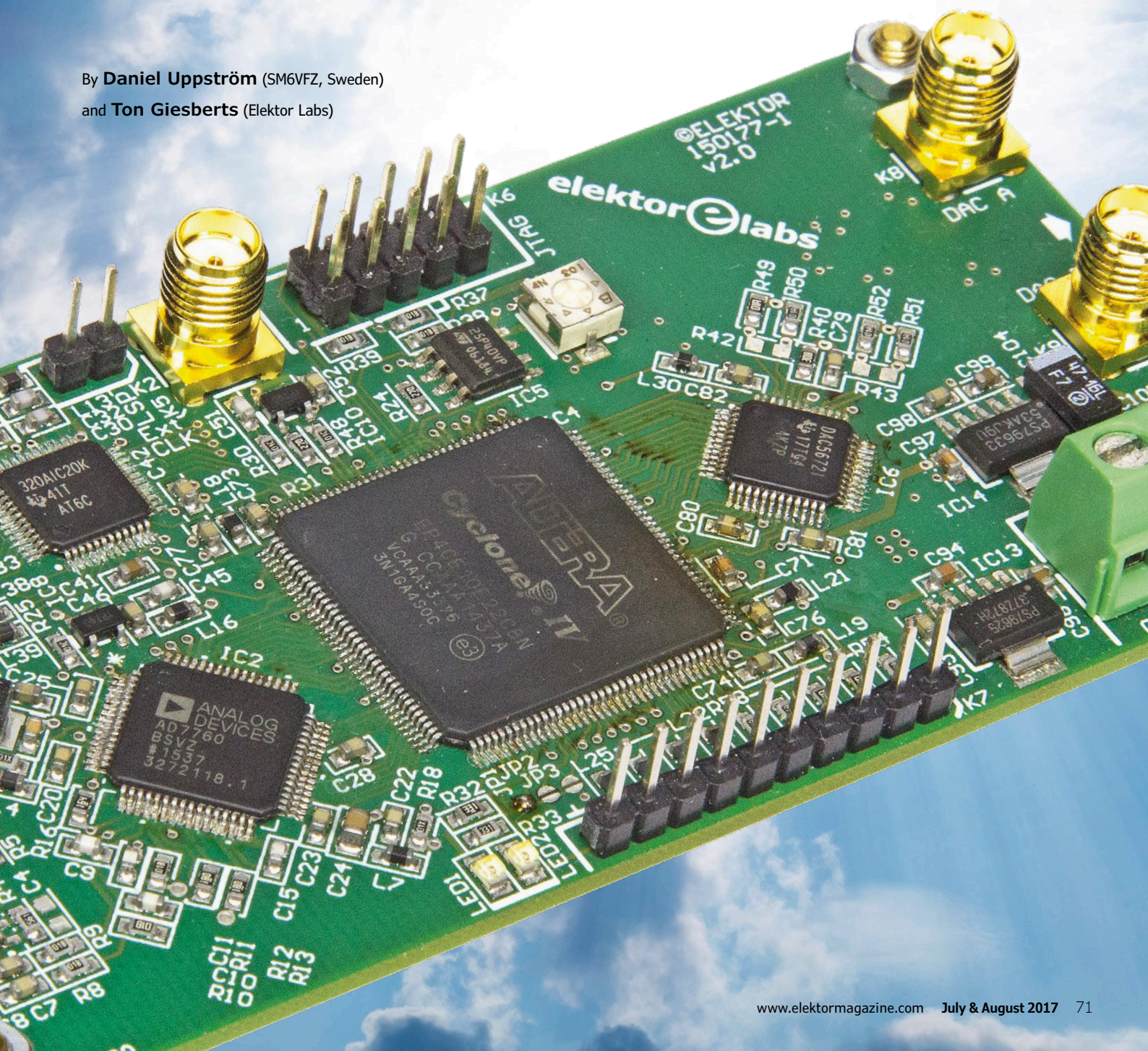


FPGA-DSP Board for Narrowband SDR

A fully programmable
receiver/transmitter baseboard

By Daniel Uppström (SM6VFZ, Sweden)

and Ton Giesberts (Elektor Labs)



Presented here is an FPGA-based digital signal processing board featuring everything necessary to do baseband processing of traditional narrowband modes like SSB/CW and AM. It solves many of the issues and limitations seen in simpler SDR platforms. Capable of transmitting as well as receiving, it provides a perfect foundation for a shortwave and/or VHF/UHF/SHF/microwave transceiver, ideal for scientific, ham, and other serious radio applications.

From the good old days...

There once was a time when radio technology was the focus for the bulk of electronic hobbyists and electronic engineers. Many people built their own equipment and having a radio amateur license was synonymous with being interested in electronics. But when the technologies evolved it became harder for hobbyists to keep up and eventually most radio amateurs bought their equipment ready-made. The state-of-the-art transceivers of the 80's and 90's made use of a very large number of complex analog building blocks and it became almost impossible for hobbyists to build something similar, not only because of the amount of knowledge needed, but also with regard to size and costs.

With the evolution of the Internet new ways of communication arose and the number of commercially-built radio transceivers for amateurs started to decrease. At the same time many compromises

were accepted in a bid to make radios smaller and cheaper while covering many frequency bands. The reduced performance with regard to essential radio properties like receiver selectivity and clean transmitted signals was to some extent compensated for by fancy digital features, color displays, and so on. However, for those who are not easily impressed by looks and gadgets and prefer a high-performance radio instead, it now again makes sense to try to build something at home.

... to software-defined radio

Still, reverting to state-of-the-art analog technology may not be the best solution and it may be more interesting to look at what is happening in the field of Software Defined Radio (SDR) where a large part of the radio signal processing is implemented in the digital domain and executed by software. With the computing power and the good analog-to-digital converters (ADCs) available today SDR technology really makes sense.

Many hobbyist and amateur radio applications process digitized signals entirely in software by a PC or a Digital Signal Processor (DSP). Although this approach is flexible, it is rather inefficient since signal processing is still relatively costly in terms of processor operations when done without hard-coded multipliers and similar peripherals. Many of the popular SDR platforms also have poor or non-existent analog filtering before digitizing, as well as low-resolution in terms of the ADC, resulting in poor performance especially with strong interfering signals around. A better way would be to build a good analog radio front-end and do the signal processing with programmable logic. Today's Field-Programmable Gate Arrays (FPGAs) available at affordable prices sport an impressive number of gates, memory bits and hard-coded multiplier blocks. They are excellent for complex signal processing tasks.

Let's build one ourselves

This project started in 2013 when the author built a first prototype and started writing VHDL-code for DSP blocks. Compared to a DSP executing code from scratch to glory, it is in general easier to implement a complex signal processing chain in programmable logic since every step can be designed independently of the others. When done, all that remains to do is connect the signals between the different blocks. Explaining this development process is beyond the scope of this article, but all the code written and all other necessary information is freely available for anyone to study [1].

Though a useful and powerful combination, programmable logic and ADCs alone do not add up to a radio. The FPGA-DSP board needs to be complemented with a radio board to do the frequency conversion, as well as with analog filtering and amplification. Such a board will be presented in a future installment of this series. A control board for tuning the radio, adjusting the volume, etc. will be the subject of yet another article. With the three boards it is possible to build a complete radio with a front panel and in a suitable enclosure.

It should be noted that the system may also be controlled by a PC connected with a USB-to-serial adapter, or a Raspberry Pi communicating over its I²C or serial port. A graphical user interface written in Python is available to demonstrate the functionality.

Circuit description

The FPGA-DSP board features a 24-bit ADC for sampling at an intermediate frequency in receive mode, a Cyclone IV FPGA for signal processing, a high-speed DAC for local oscillator and transmit signal generation, an audio interface for microphone and loudspeaker, a very stable high-precision oscillator and an I²C or UART interface for a host controller (**Figure 1**).

The heart (or brain) of the FPGA-DSP board is the EP4CE10 Cyclone IV FPGA from Intel, formerly Altera (IC4). This chip can be configured for virtually any digital function. Governed by the firmware, at start-up its gates are configured by an external memory (IC5) known as the configuration memory.

For audio input and output there is a so-called CODEC (coder-decoder) — a TLV320AIC20K (IC3) — comprising two ADC/DAC channels with 16-bit reso-

PROJECT INFORMATION

FPGA DSP SDR

Radio Transmitter

Receiver Ham Radio

entry level

intermediate level

→ expert level

8 hours approx.

SMD soldering tools,

JTAG programmer (USB Blaster), Quartus Lite

€190 / €200 / \$210 approx.

lution and a maximum sample rate of 25,000 samples per second (SPS). It has a built-in digital 8-kHz lowpass filter, besides a microphone amplifier and a speaker driver that can deliver up to 250 mW into 8 Ω. All audio inputs and outputs sport programmable gain/attenuation. The speaker gets connected to K2, the microphone to K3; all other channels available in IC3 are connected to K4, making them available for auxiliary equipment.

The main input of the board is through K1; it expects a differential signal, typically in the hundreds of kHz range. The signal passes a differential amplifier (IC1) and a discrete lowpass filter, to be digitized finally by 24-bit ADC IC2, a type AD7760. This chip has many power supply connections separated by supply voltage and/or passive filters. Its master clock is provided by the FPGA, level-shifted to 5 V by IC8.

There is also a fast two-channel digital-to-analog converter (DAC) on the board, a DAC5672 (IC6). Its two differential outputs A and B are converted with transformers to single-ended outputs; low-pass filtering is provided, allowing signals up to 50 MHz to be produced. The two DAC outputs are available at SMA-type coaxial connectors K8 and K9. The board's main clock is generated by IC7, a 20-MHz Temperature-Compensated Crystal Oscillator (TCXO). The clock signal is fed to the FPGA which distributes it to the peripheral ICs. The frequency can be fine-tuned with potentiometer P1. Connector K5 provides an input for an external reference source — if one happens to be available. If so, the TCXO can be switched off by means of T1. The TCXO and the external reference signals are both guided to the FPGA through unbuffered inverters wired as analog amplifiers (IC9, IC10) and consequently may be of moderate voltage swing.

The clock is divided down in the FPGA to generate a low frequency signal for LED1 indicating that the clock is running and the FPGA is configured. The function of LED2 is reserved for future needs. The board's 5-volt supply voltage is connected to K10 from where it is distributed to four different Low Drop-Out (LDO) voltage regulators (IC11-14) to create the supply voltages — 1.2 V, 1.8 V, 2.5 V and 3.3 V — required for various components on the board.

The interface to the host controller con-

Features

- FPGA + DSP + Audio CODEC
- Narrowband software defined radio platform
- Can be used for radio operation at virtually any frequency between zero and many GHz
- Uses superheterodyne principle
- Low second IF higher than 0 Hz
- Receive (RX) and Transmit (TX) compatible
- Weaver SSB modulator/demodulator

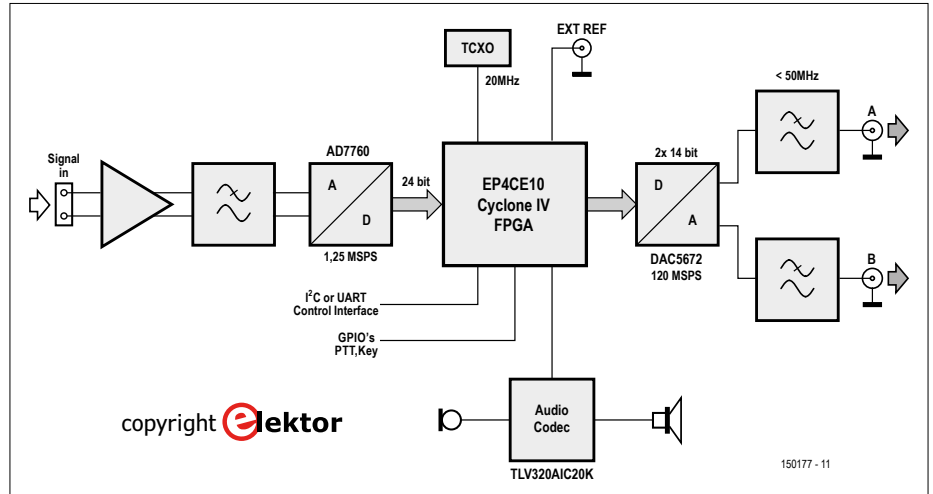


Figure 1. Block diagram of the FPGA-DSP board.

sists of two pins intended for I²C or UART communication, selectable by solder jumper (JP1). They are available on K7, together with seven additional I/O lines intended typically for PTT (push-to-talk) and Morse key signals. These pins might also be used for I²S audio I/O. Solder jumpers JP2 and JP3 do not have a function in the current FPGA firmware. K6 provides a JTAG interface for programming the FPGA and its configuration memory.

Radio topology

With a suitable radio board, the FPGA-DSP board can be used for radio opera-

tion at virtually any frequency between zero and many GHz. The typical applications, however, are for shortwave (under 30 MHz) and/or the 2-m amateur band (144-146/148 MHz).

For shortwave reception, a simple radio board could be constructed as shown in **Figure 3**. The antenna signal is lowpass filtered and amplified. By mixing it with a local oscillator (LO) signal from DAC A, the frequency of interest is converted to 45 MHz before it is passed through a crystal filter. This intermediate frequency (IF) signal is amplified before entering a second mixer where it is converted down to a second IF signal centered at

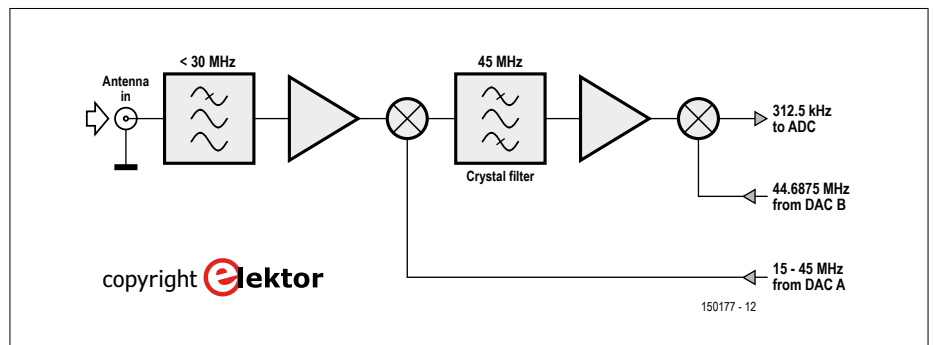


Figure 3. Block diagram of a radio add-on board.

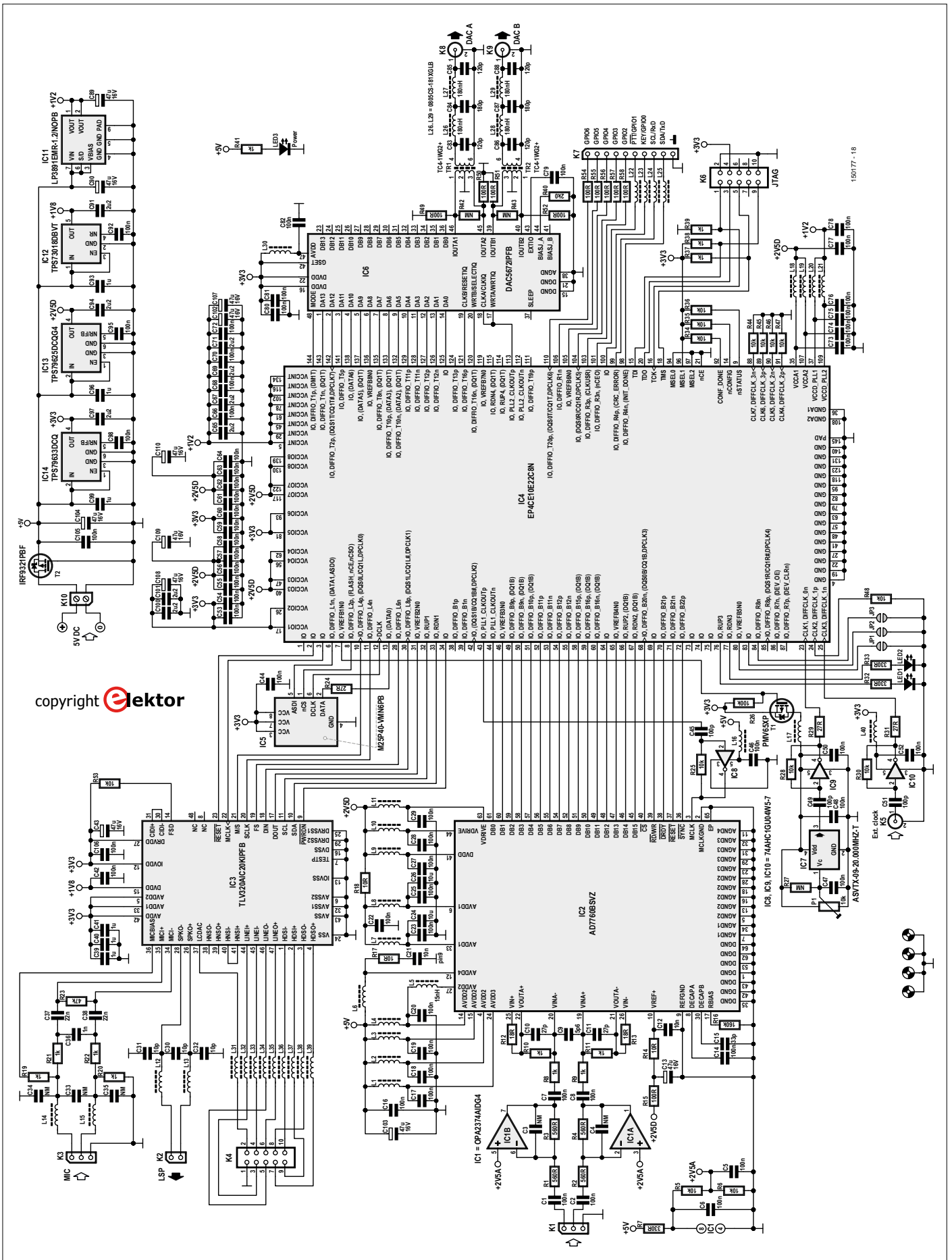


Figure 2. If you manage to look through the mist of passive filters, decoupling capacitors, data buses and supply lines, what remains is basically a four-chip schematic!

312.5 kHz (see below) with the help of a second LO signal provided by DAC B. The second IF signal is digitized by the ADC and fed into the FPGA where it is processed and ultimately demodulated to audio.

Compared to many other SDR-solutions this topology is advantageous for two reasons:

1. Double conversion superheterodyne principle.

The signal is mixed to a high-frequency first IF where it is filtered. Then it gets converted down to a low-frequency second IF and subsequently, sampled. This is advantageous since strong signals on frequencies other than the one of interest are filtered out by the narrow filter positioned early in the chain. This implies high selectivity, or high dynamic range, allowing the receiver to receive weak signals in the presence of strong ones. The simpler SDR projects often use the 'zero-IF' principle instead, with only one mixer, no filter, and simple digitizing at audio frequencies around the carrier. The approach is limited in its ability to handle strong interferers, since mixing products of unwanted signals are likely to fall in the audio band where they will distort the signal you want to hear.

Fancier SDR solutions simply sample a large portion of the frequency spectrum ('direct sampling') and do the tuning in the digital domain. This method creates interesting opportunities for monitoring large chunks of the spectrum, but, when combined with a desire for good selectivity and/or dynamics, it places very strict requirements on the linearity of the input amplifier and resolution of the ADC in terms of effective bits. Fast sample rates and high resolutions usually imply high costs and power demands, often resulting in poor compromises.

2. The second IF is low but not zero.

This means that an interferer within the filter bandwidth (typically 15 kHz) cannot mix to produce a frequency interfering with the received signal. These kinds of mixing products will be sampled but digitally filtered out afterwards because they end up below the desired signal centered at 312.5 kHz. Since this frequency is not too high, it requires only a sample rate in the range of 1-2 MSPS for which one can find affordable 24-bit ADCs. These 24 bits result in a digital dynamic range of more than 100 dB and eliminate the

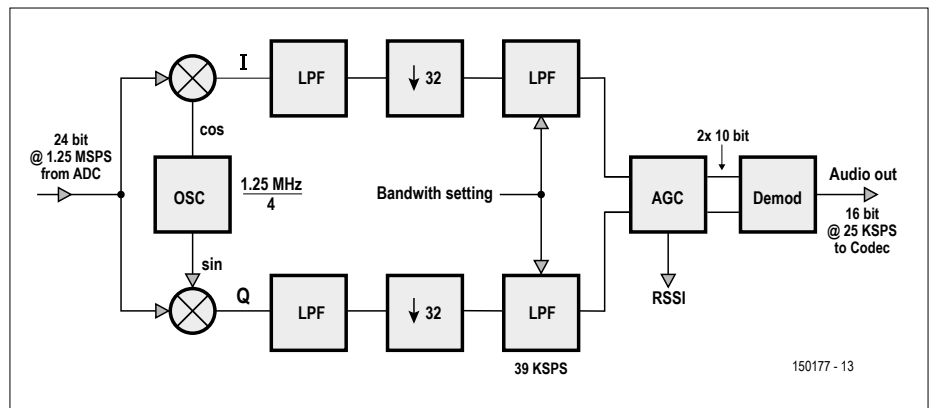


Figure 4. The signal processing chain implemented in the FPGA.

need for gain control in the analog blocks, which, in turn, simplifies the receiver circuitry a lot.

Digital signal processing — reception

It is the FPGA's job to process the sampled input signal. **Figure 4** shows a simplified diagram of the implemented blocks. The ADC digitizes the input signal centered at 312.5 kHz at a rate of 1.25 million samples per second (MSPS). With 24 bits per sample at 1.25 MSPS there are 30 Mbits/s to process. This is a lot more than necessary and so the first digital processing steps are to down convert and down sample the input stream. By multiplying the signal with two sequences [0,1,0,-1,0,...], a "sine wave" at exactly a fourth of the sample rate, and [1,0,-1,0,1,...], a cosine (the second sequence is identical to the first,

but shifted one position or 90°) the signal at 312.5 kHz is down-converted to a quadrature signal centered at 0 Hz. These signals are then fed to down-sampling low-pass filters that reduce the sample rates by 32 — simply by throwing away 31 out of 32 samples —, to provide outputs at sample rates of around 39 KSPS. These two down-sampled signals now each contain information from zero to about 10 kHz; combined, they represent the information that was within ±10 kHz of 312.5 kHz in the sampled input signal. In case you didn't guess it, the second IF of 312.5 kHz was chosen because it is a quarter of the 1.25-MHz sample rate. This allowed the implementation of the down converter in VHDL to be kept simple because multiplying by 1, 0, and -1 is easy (**Listing 1**). Other IF frequencies would have required a sine table and 24-bit multipliers.

Listing 1. VHDL implementation of a down converter at 1/4 of the sample rate.

```

if ns = 0 then
Ia(write_pointer) <= signed(Data_in); -- 1
Qa(write_pointer) <= to_signed(0,24); -- 0
ns := 1;
elsif ns = 1 then
Ia(write_pointer) <= to_signed(0,24); -- 0
Qa(write_pointer) <= signed(Data_in); -- 1
ns := 2;
elsif ns = 2 then
Ia(write_pointer) <= (not signed(Data_in)) + 1; -- -1
Qa(write_pointer) <= to_signed(0,24); -- 0
ns := 3;
elsif ns = 3 then
Ia(write_pointer) <= to_signed(0,24); -- 0
Qa(write_pointer) <= (not signed(Data_in)) + 1; -- -1
ns := 0;
end if;

```

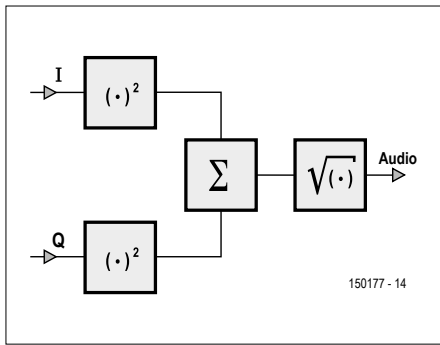


Figure 5. AM signal demodulation.

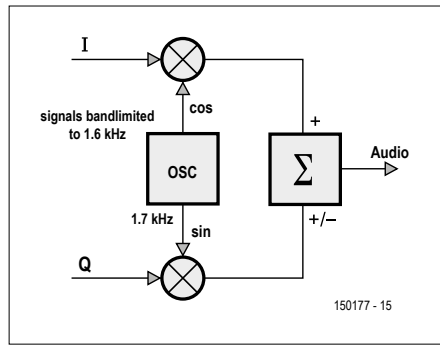


Figure 6. SSB and CW signal demodulation.

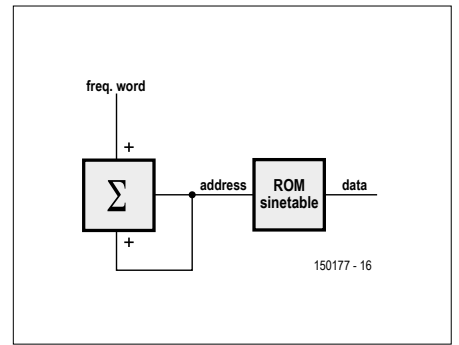


Figure 7. The basic operating principle of DDS. The block with the sigma symbol in it is the phase accumulator.

The next step is to lowpass-filter the two signals once more. This time the cutoff frequencies are adjustable in order to match the selected mode of operation. For AM, their cutoff frequencies are at 5 kHz for 10 kHz of information bandwidth. For SSB (Single-Side Band) telephony, they are set to around 1.6 kHz (3.2 kHz bandwidth), and for narrow band CW (Continuous Wave; Morse code) they filter at only a few hundred hertz. Two times 24-bit resolution still yields a lot of data to process. To extract the information of interest, i.e. to demodulate, we do not need all these bits. The task of the next block, the Automatic Gain Control (AGC), therefore is to continuously monitor the signal strength and scale the signal so as to output only two 10-bit streams that can be demodulated. The AGC block also outputs the Received Signal Strength Indication (RSSI). Next is the demodulator. For AM signal demodulation this amounts to the extraction of the amplitude by

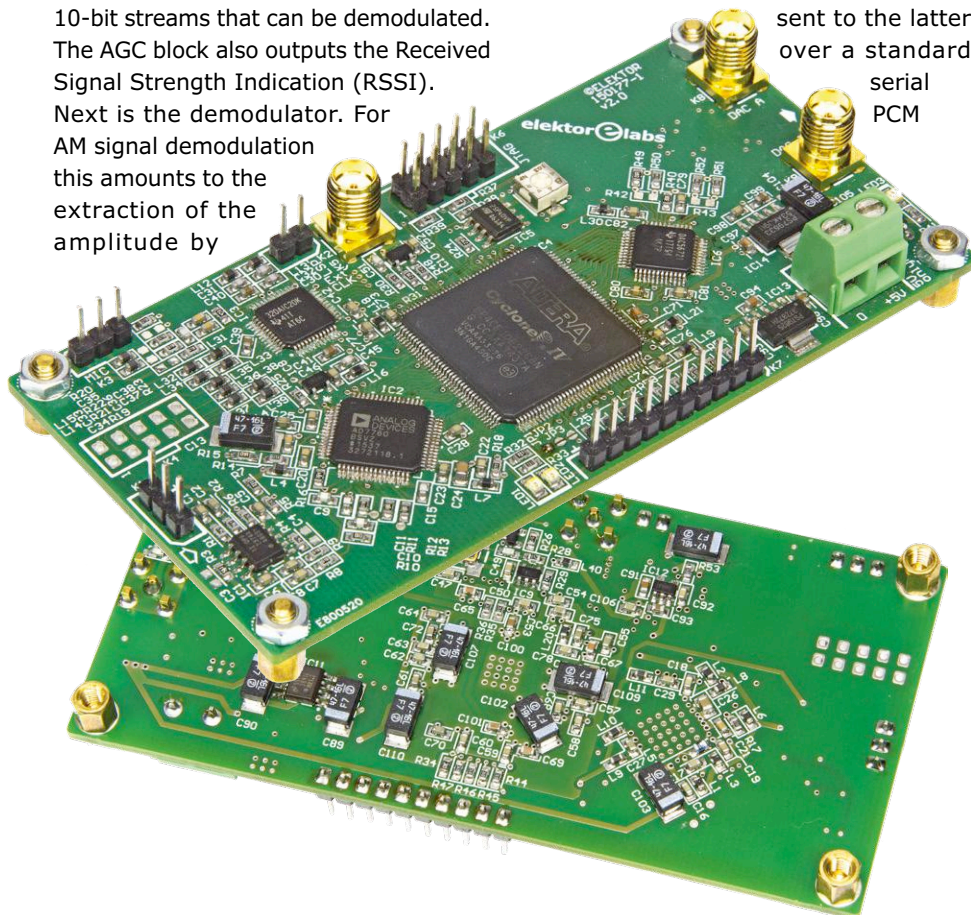
using the Pythagorean theorem for two orthogonal vectors: $\sqrt{I^2+Q^2}$, see **Figure 5**. For SSB (and CW) demodulation is unfortunately a bit more complicated. In this design Weaver's method was used [2][3]. The main idea (**Figure 6**) is to split the signal bandwidth in two and mix again with a locally generated low frequency quadrature signal in the middle of the band. Choosing between Upper Side Band (USB) or Lower Side Band (LSB) is then just a matter of sign manipulation in the final adder.

The demodulated audio finally passes a lowpass filter to remove any high-frequency components caused by distortion in the AGC or demodulator. It is then resampled to match the sampling rate of the audio CODEC and sent to the latter over a standard serial PCM

interface. The audio CODEC provides programmable gain and can drive a speaker or headphone directly. The local oscillator (LO) signals at the outputs of the two DACs are produced by two Direct Digital Synthesis (DDS) blocks inside the FPGA. Such a synthesizer consists of a so-called phase accumulator register whose value is increased by a fixed amount at every clock cycle of the 120-MHz DAC clock (which is derived from the internal PLL of the FPGA), thus the increase determines the frequency. The accumulated value is used as an address into a sine table; the value found at that address is sent to the DAC (**Figure 7**). The frequency word for DAC A, which forms the tunable LO, is 25-bits wide. This corresponds to a frequency step of $120 \times 10^6 / 2^{25} = 3.57$ Hz. In addition, there is a fine-tune mechanism, implemented with fractional values, giving a final resolution of 0.44 Hz. For the second LO, which is a fixed frequency for conversion between the first IF and the second, the corresponding DDS is implemented with a lower resolution to save FPGA resources — it has a precision of about 38 Hz.

FPGA programming

The compiled firmware for this project is available as a 'JTAG indirect configuration' file (trx.jic) that first configures the FPGA as a bridge and then programs the configuration memory through the FPGA. This is done with the aid of a USB Blaster adapter that connects to a PC over USB to form a JTAG interface to the board. Such adapters can be found for little money on popular shopping websites. The software to use at the PC side is known as Quartus Lite and can be freely downloaded from the Intel/Altera website. If the complete development envi-





COMPONENT LIST

Resistors

Default: 1%, 100mW, 0603

- R1,R2,R3,R4 = 560 Ω
- R5,R6,R25,R28,R30,R34,R35,R36,R44-R47,R48,R53 = 10k Ω
- R7,R32,R33 = 330 Ω
- R8-R11,R19-R22,R37,R38,R39,R41 = 1k Ω
- R12,R13 = 18 Ω
- R14,R17,R18 = 10 Ω
- R15,R49-R52,R54-R58 = 100 Ω
- R16 = 160k Ω
- R23 = 47k Ω
- R24,R29,R31 = 27 Ω
- R26 = 100k Ω
- R27,R42,R43 = not fitted
- R40 = 2k Ω
- P1 = 10 k Ω trimpot, 4.5mm, SMD

Capacitors

Default: 5%, 0603

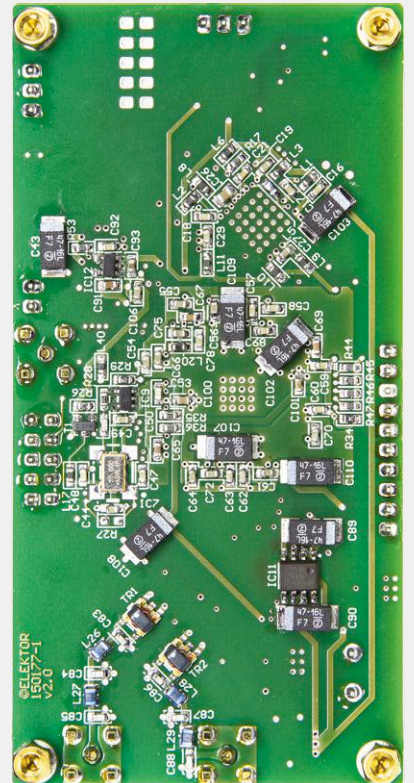
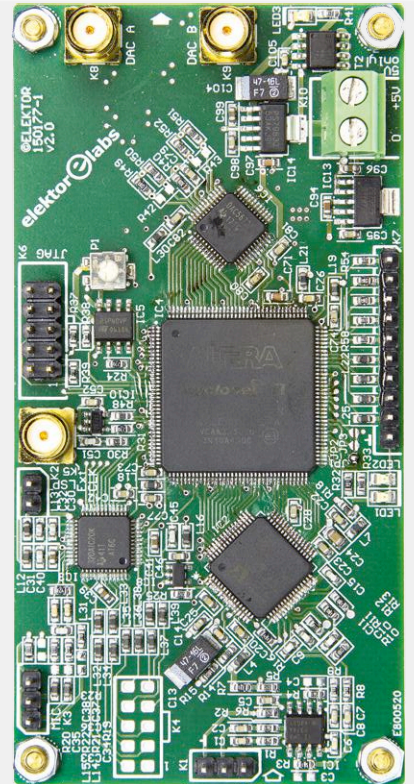
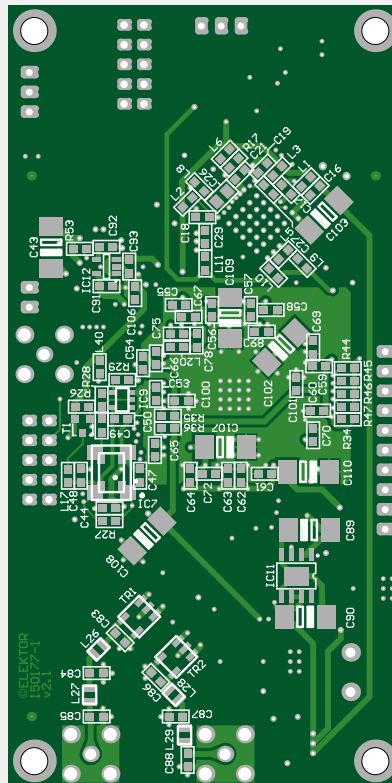
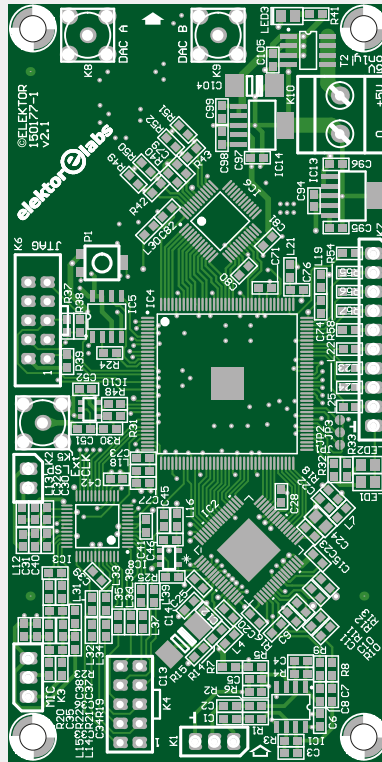
- C1,C2,C5-C8,C14,C16-C19,C20,C22,C23,C25,C27,C28,C29,C42,C44,C46,C47,C48,C50,C52,C53-C64,C66,C68,C70,C72-C82,C92,C95,C98,C105,C106 = 100nF
- C3,C4,C33,C34,C35 = not mounted
- C9 = 5.6pF \pm 0.25pF
- C10,C11 = 27pF, 1%
- C12,C21 = 10nF
- C13,C43,C89,C90,C102,C103,C104,C107-C110 = 47 μ F 16V, tantalum, 0.35 Ω , SMD Case C
- C15 = 33pF
- C24,C26 = 10 μ F 6.3V, 0805
- C30,C31,C32 = 10pF
- C36 = 1nF
- C37,C38 = 22nF
- C39,C40,C41,C93,C96,C99 = 1 μ F, X7R
- C45,C49,C51 = 100pF
- C83,C85,C86,C88 = 120pF
- C84,C87 = 180pF
- C65,C67,C69,C71,C91,C94,C97,C100,C101 = 2.2 μ F, X7R

Inductors

- L1-L4,L6-L25,L30-L40 = 1k Ω @ 100MHz, 200mA, 0603
- L5 = 15nH, 5%, 170m Ω , 700mA, f_{res} 4GHz
- L26-L29 = 180nH, 2%, 640m Ω , 400mA, 0805
- TR1,TR2 = TC4-1WG2+

Semiconductors

- IC1 = OPA2374AIDG4
- IC2 = AD7760BSVZ
- IC3 = TLV320AIC20KIPFB
- IC4 = EP4ACE10E22C8N
- IC5 = M25P40-VMN6PB
- IC6 = DAC5672IPFB
- IC7 = 20MHz crystal oscillator, adjustable, 5 \times 3.2mm
- IC8,IC9,IC10 = 74AHC1GU04W5-7
- IC11 = LP3891EMR-1.2/NOPB



- IC12 = TPS73018DBVT
- IC13 = TPS79625DCQG4
- IC14 = TPS79633DCQ
- LED1,LED2,LED3 = green, 0805
- T1 = PMV65XP
- T2 = IRF9321PBF

Miscellaneous

- K1,K3 = 3-pin pinheader, 0.1" pitch
- K2 = 2-pin pinheader, 0.1" pitch

- K4 = 14-pin pinheader, 0.1" pitch
- K5,K8,K9 = SMA straight jack, 50 Ω
- K6 = 10-pin (2 \times 5) pinheader, 0.1" pitch
- K7 = 10-pin pinheader, 0.1" pitch
- K10 = 2-way PCB screw terminal block, 0.2" pitch
- PCB # 150177-1



Figure 8. The prototype built by the author doing USB at 144 MHz.

Web Links

- [1] www.elektormagazine.com/150177
- [2] Donald K. Weaver Jr., "A Third Method of Generation and Detection of Single-Sideband Signals", www.h4.dion.ne.jp/~ja5fp/weaver.pdf
- [3] Daniel Uppström, "Weavers metod för SSB", ESR Resonans 2/2014, http://resonans.esr.se/ESR_Resonans_2014_2.pdf (in Swedish)
- [4] www.elektormagazine.com/labs/fpga-dsp-radio-for-narrow-band-communications-150177-i
- [5] <https://github.com/ast/dsp-sdr>

ronment is not of interest there's always the option to download the applications needed for programming only. The Quartus suite is available for both Windows and Linux.

Control interface

At power on, the firmware gets loaded from the configuration memory, and programmed into the FPGA. Then the FPGA has to be initialized and controlled over I²C or the serial port. The FPGA is much like a typical IC with a few registers that need to be set. Every data transmission involves five bytes being sent to the FPGA. The first two bits set the register address while the remaining 38 bits contain configuration data.

A few status signals can be read as well as the value of the received signal strength indication (RSSI). The controller typically polls this information a few times per second to update signal strength indication to the user.

A detailed description of the control interface, with full register map, is available together with the firmware and its source files in a GIT repository, accessi-

ble through the project web pages [1], [4], and [5].

In a standalone radio the controller is likely to be a small microcontroller board with a display and buttons (like the Elektor Platino). A suitable board will be described in a subsequent installment. For experiments, or when the need for a monitor and mouse/keyboard is not an obstacle, a Raspberry Pi should also make a flexible control interface. For the RPi (or any other Linux computer), an applet was written in Python using the GTK graphical framework, which makes the controls easily accessible. By default this applet connects to the FPGA board over the serial port. It also has an experimental socket mode to enable remote operation. The idea is to run a server on a Raspberry Pi at the location of the radio and connect to the Pi with the applet from a remote computer. With the large amount of man-made noise in urban areas, and with the limitations and regulations against installing large antennas in these locations, the option of remote operation becomes increasingly interesting for radio amateurs and

shortwave listeners.

It should also be possible to route the audio from the FPGA directly to the PCM/I²S ports of the Raspberry Pi in order to avoid the detour over analog audio when doing remote operation. This is yet to be tested.

How to download the applet, install necessary packages and connect the Raspberry Pi to the FPGA board is described in a document available with the source code.

Conclusion

The project presented in this article is a powerful building block for radio amateurs and experimenters who want to build their own radios without compromising on performance. At the same time it is hoped that this project will inspire the more digitally oriented readers to dive into the world of radio and signal processing.

The project has a lot of room for improvements and additions. A future installment (hopefully in the next edition) will present a radio board and discuss the transmission side of things; if enough resonance is observed, we may present more advanced radio boards.

The VHDL code for the FPGA is available as open-source on [5] and additions and improvements are welcome.

Finally, it should also be noted that the board is generic enough to be used as a development board for other FPGA-DSP applications.

Please visit the project web page for updates and additions. ◀

(150177)

FROM THE STORE

→ 150177-1:
Bare PCB for FPGA-DSP board

→ 150177-91:
Ready-built FPGA-DSP board

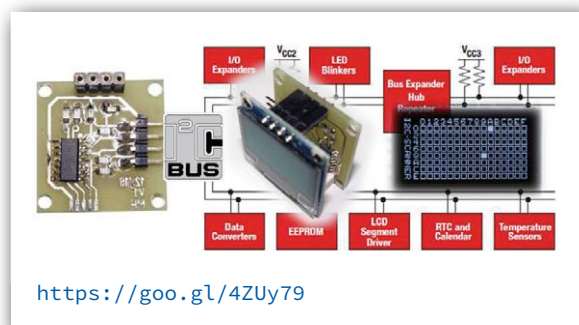
Elektor Labs Pipeline



A large part of electronics is all about measuring something. This time we present a few projects you can build yourself to measure heat, waste, time and I²C addresses.

I²C bus scanner has OLED display

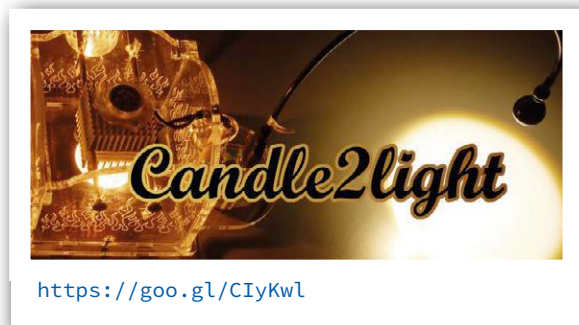
The Inter Integrated Circuits bus, better known as I²C bus, was developed by Philips in the 1980s to connect a microprocessor or microcontroller in an easy manner to other integrated circuits in — at the time of invention — mostly television sets made by the brand and its sub-brands. Little by little the bus became popular and today components with I²C interface are all over the place. Because the bus pops up everywhere, tools are needed to check if it is working correctly. The scanner proposed here helps you to quickly see if the connected devices are actually alive.



<https://goo.gl/4ZUy79>

Candle2light — a luminous efficacy booster

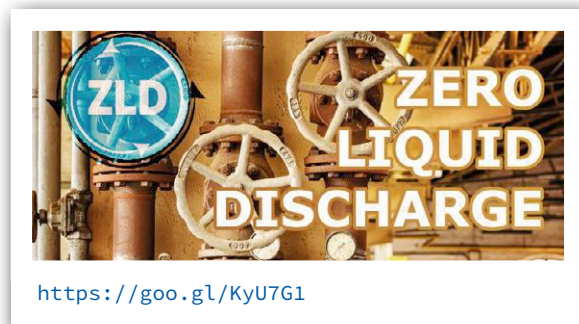
Before electric lighting became common in the early 20th century, candles and oil lamps were commonly used for illumination. Even a modern candle producing approximately 80 W of heat energy has a luminous efficacy almost a hundred times lower than an incandescent light bulb. Tea lights are even worse. With a Peltier element it is possible to convert the heat released by a tea light into electricity and drive a highly efficient LED reading light, allowing you to read your favorite book by electric candle light.



<https://goo.gl/CiYkwl>

Zero liquid discharge monitoring with LoRaWAN

Zero liquid discharge (ZLD) is one major step towards making an industrial plant non-polluting. It means that a plant can take in water but it should not discharge any wastewater. ZLD plants produce solid waste. This project provides a cheap, solar-powered wireless liquid level monitor for plants that cover a large surface and that may have multiple drains. An ultrasonic transceiver, together with an Arduino and a LoRaWAN module is about all that is needed at the drains, plus a laptop running Google maps at the office side of things.



<https://goo.gl/KyU7G1>

So retro, but oh so stylish: the wooden Nixie watch

For many years now I am in the possession of a stylish, shiny wooden attaché case, brought home from a trip to Tanzania. I never use it though, because unfortunately I totally lack style. Maybe building and wearing this beautiful wooden watch with its retro Nixie tube display improves matters a bit? Maybe start wearing wooden glasses too? Or maybe I just should stop wearing wooden shoes? ◀

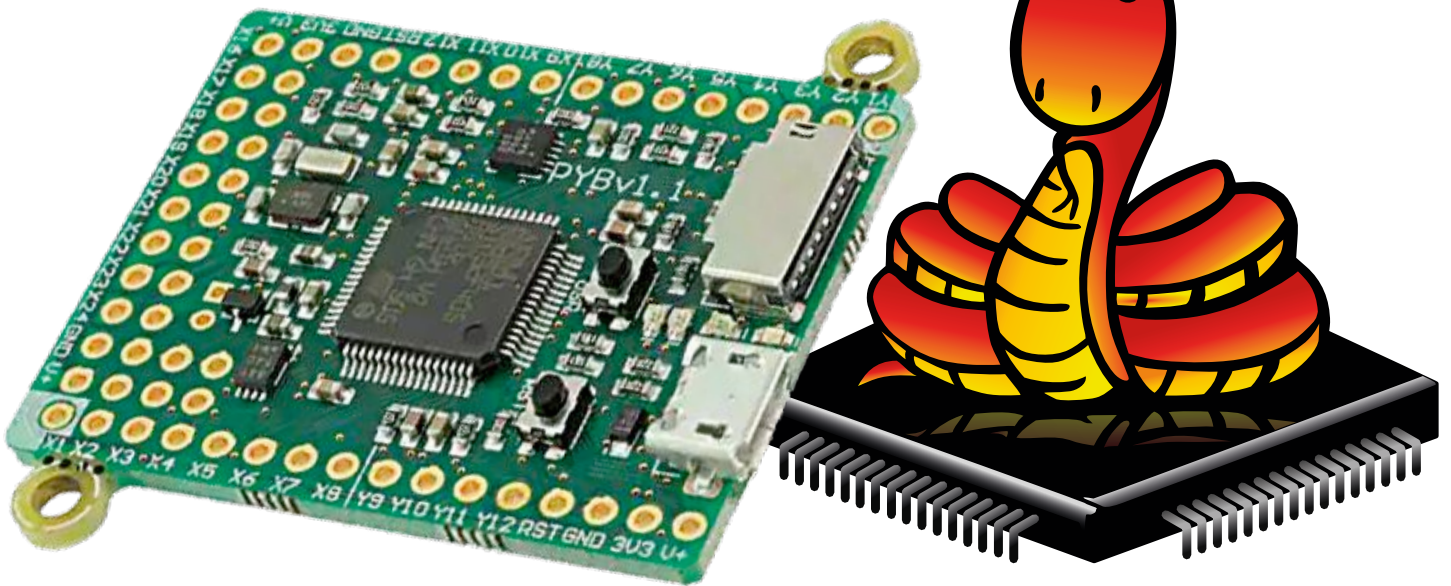
(160401)



<https://goo.gl/lUMZRU>

MicroPython and the pyboard

From blinking LED to ... webserver blinking an LED



By **Clemens Valens** (Elektor Labs)

Since its first release in 1991, the Python programming language got widely adopted, and today it has become the preferred language of many programmers. Although intended for PC applications, a few years ago Python got ported to embedded systems too. Called MicroPython (uPy), it is a subset of Python running on platforms like the BBC micro:bit and the ESP8266. And hey, it also runs on the pyboard, the official uPy demonstrator board.

The Micro Python project (written as two words at the time), including the pyboard, was on Kickstarter by the end of 2013 where it scooped up more

than six times the initial funding goal of £15,000. This success implied that the team had to throw in support for the CC3000 Wi-Fi module, the WIZ820i0

Ethernet module, and the NRF24L01+ low-power wireless module. Now, almost four years later, it's time to see what has come of it all. Let's have a play with the pyboard and uPy.

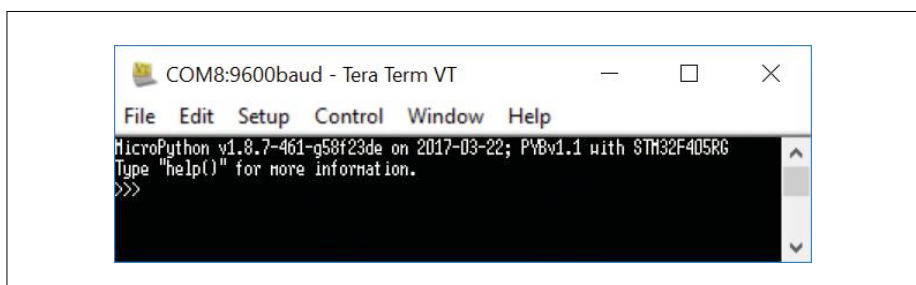


Figure 1. The REPL.

Getting started with the pyboard

In front of you, you should have a pyboard v1.1 (PYBv1.1, nicely packaged), a micro USB cable and a computer (I used a Windows 10 laptop). Connecting the board to the computer immediately brings up a window showing the contents of drive E, labeled PYBFLASH. If you happened to have an SD card

plugged in, its contents will be shown instead. The operating system should also find a serial port. After launching a serial port terminal program (like Tera Term), connecting it to the pyboard's port and pressing a key, you should see the Python prompt (**Figure 1**):

```
MicroPython v1.8.2 on 2016-07-13;
PYBv1.1 with STM32F405RG
Type "help()" for more information.
>>>
>>>
```

That was fast, wasn't it? Less than a minute after plugging in the board you are in business — a minute mostly spent on moving the mouse around to open and navigate the operating system's menus.

REPL

What you are now looking at is a simple, interactive programming environment that takes single expressions, evaluates them, and returns the result. Such an environment is known as a Read-Evaluate-Print Loop or REPL. It is important to know this acronym, because, when doing Python, you will encounter it a lot. People that have worked with BASIC will understand this kind of interface, although it wasn't probably called REPL when they used BASIC. Python is, like BASIC, an interpreted programming language, and, like BASIC, Python can be compiled to make it run faster. As a matter of fact, Python is a lot like BASIC.

Next step: blink an LED

As the Python prompt suggests, type 'help()' and press the enter key. This brings up a list of possible commands to get you started (**Figure 2**). Having a close look at this list is interesting as it reveals what is possible with the board. Built around an STM32F405RG ARM Cortex-M4F microcontroller, the board apparently has a real-time clock (RTC), an analog-to-digital converter (ADC), a digital-to-analog converter (DAC), four LEDs, switches, an accelerometer, a random number generator (RNG), I²C, SPI, UART, I/O pins and it supports servos. Some functions that control these peripherals are listed too, making for a pretty comprehensive help indeed. Simply looking at the board also reveals an SD card connector.

At the prompt, type the command below and press <Enter>.

```
>>> pyb.LED(1).on()
```

Notice how a red LED, next to the 'USR' pushbutton, lights up. Repeating the command but with the '1' replaced by a '2' (use the arrow keys to go back to the previous command and edit it) lights up a green LED next to the red one. LED number 3 is yellow, number 4 is blue. Except for the yellow one, the LEDs are incredibly bright and you better not look right into them. Let's continue therefore with LED(3), leaving you with the exercise to switch off the other three LEDs. To blink an LED forever a loop is needed, for instance the `while` loop. To make it loop forever use a condition that will always be true, like the Boolean constant `True`. Enter the command (don't forget the colon ':' at the end)

```
>>> while True:
```

Next, enter a toggle LED command. The terminal will start indenting the code as is mandatory in Python, shown by three dots. Enter a delay command to set the blink rate, 250 ms for instance. You should now have something like this:

```
>>> while True:
...     pyb.LED(3).toggle()
```

Features

- Programming in MicroPython
- Simple webserver
- 100% Arduino-free

PROJECT INFORMATION

- Python, MicroPython, pyboard, webserver
- entry level, intermediate level, expert level
- 2 hours approx.
- pyboard v1.1, PC, serial terminal
- £30 / €35 / \$40 approx.

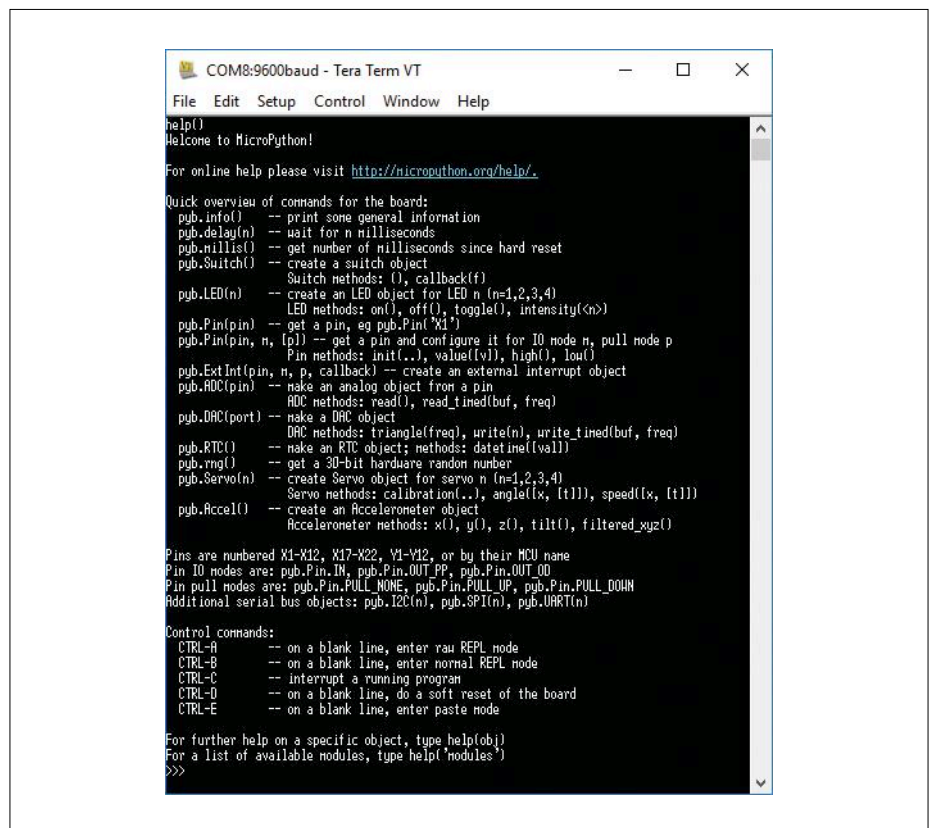


Figure 2. MicroPython also contains a comprehensive help.

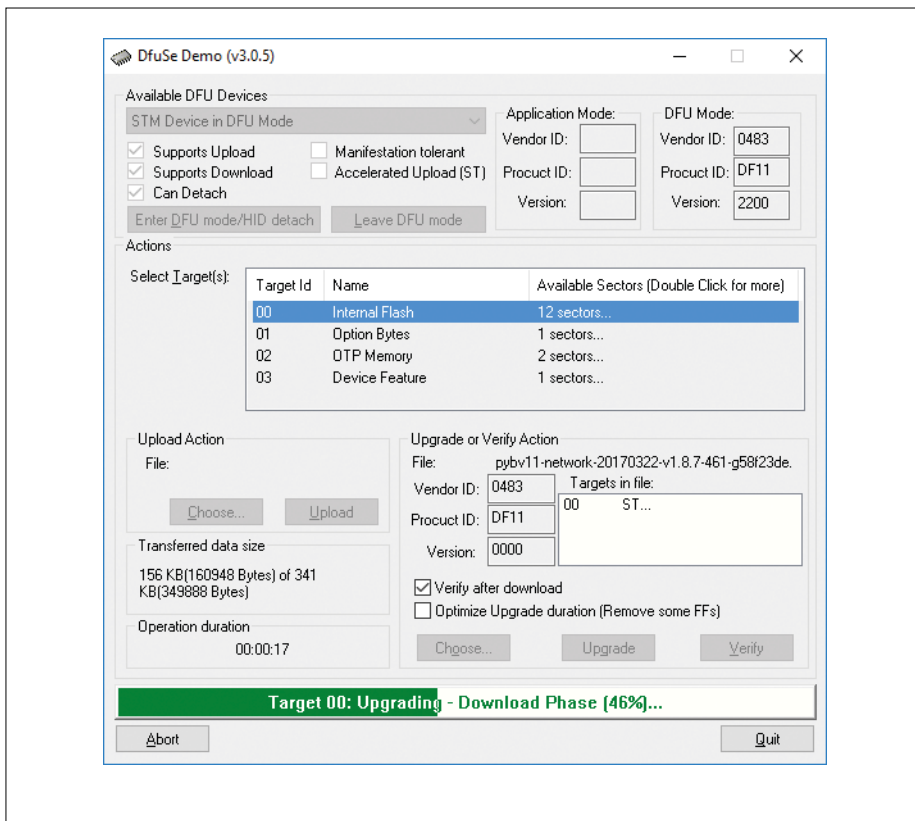


Figure 3. DfuSe busy writing new firmware to the pyboard.

```
...     pyb.delay(250)
...
-1 -2 21
-1 -2 20
-2 -1 20
...
```

To get this loop going you have to leave indentation mode first by pressing the backspace key, then press enter. LED3 should now blink at the rate you entered. As you may have noticed, the terminal does not accept commands while the loop runs. To break out of the loop and return to the command prompt, press Ctrl-C ('interrupt a running program'). Now all the commands that were (accidentally) entered while the loop was running will be executed, so better be careful what you do during program execution!

pyboard accelerometer

Here is a quick experiment to see if the board's accelerometer is working correctly.

```
>>> while True:
...     print(pyb.Accel().x(),pyb.
Accel().y(),pyb.Accel().z())
```

The output should be an endless list of the three values x, y and z. Rotate, spin, flip, roll, jaw, and shake the board to make the numbers change.

As before, press Ctrl-C to return to the prompt. This is a revealing test as it shows the speed of execution of an apparently simple loop. All it does is read a sensor and print three integer values. I measured about three lines per second, which is not very fast indeed.

The reason for this slowness is that the program is interpreted on-the-fly instead of being transformed (compiled) into machine instructions first, and even though there is only one line of code (ignoring the while statement), this is in reality a quite complex instruction. It is possible to speed things up by creating so-called frozen modules, but doing this implies recompiling the complete firmware for the pyboard, which sadly is outside the scope of this article.

My first program

After playing a while in the REPL you may be wondering how to transform your volatile experiments into something more, let's say, persistent. This is easy enough; all you need is a good text editor.

The pyboard is accessible as a flash drive and when you open it in a file viewer, you will see the file `main.py`. This is the Python file that is executed after a pyboard reset (hard with the 'RST' push-button, or soft with the Ctrl-D command). Open this file in the text editor and add your commands to it. When done, save the file and reset the board to launch your program.

Getting out of a (deep) hole

Don't be afraid to get creative, trial and error is the way to go. If you manage to mess things up in such a way that nothing seems to be working anymore, know that there are two rescue protocols that may help you out: safe mode and factory reset. In safe mode the files `boot.py` and `main.py` are not executed, giving you access to the file system (the USB drive should appear). Now you can edit them to fix any problems. Entering safe mode starts by pressing the 'USR' button. Hold it down while you briefly press the 'RST' button. Now watch the LEDs. When only the yellow LED is on, release the 'USR' button. The board is now in safe mode.

Factory reset goes a step further than safe mode. It deletes all the files on the internal pyboard storage (not those on the SD card), and restores the files `boot.py`, `main.py`, `README.txt` and `pybcdc.inf` back to their original state. To do a factory reset, proceed in the same way as for entering safe mode. However, this time, when the LEDs are doing their little dance, release the 'USR' button when both the yellow and the green LEDs are on. The yellow LED flashes quickly three times and you are back to where you started: the beginning of this article.

If the above methods did not help you solve your problems, you have one last resort: reprogram the firmware. For this you must put the board into Device Firmware Update (DFU) mode. This is achieved by shorting pin P1 to 3V3 (with a paperclip) before resetting the board by pressing the 'RST' button. The first time you do this you may have to install a DFU driver first. If Windows does not manage to do this all by itself, as was the case on my computer, you will have to help it a bit. Download the package STSW-STM32080 (or DfuSe Demo) from the STMicroelectronics website and install it. Point Windows to `<installation path>\STMicroelectronics\Software\DfuSe v3.0.5\Bin\Driver\`

to make it install the driver. If all went well, DfuSe Demo should now list the board (**Figure 3**).

Precompiled firmwares can be found on the uPy website. Make sure to pick one that corresponds to your board. In the 'Upgrade or Verify Action' area (not the 'Upload Action' area) click 'Choose', navigate to the DFU file that you want to program and click 'Upgrade' (not 'Upload'). Close DfuSe, remove the boot jumper and press 'RST'. You may now have to do a factory reset to restore the flash file system. In my case it was necessary to let Windows repair the pyboard drive after a firmware update.

DFU is, of course, also possible on Linux and Mac OS by using dfu-util or pydfu.

Not all mistakes are yours

When you start to learn writing uPy programs you will, of course, encounter your share of errors and mistakes. One of the most common — and frustrating — errors is forgetting to use the tab key. Or thinking that you were using tabs, when in reality you were not. Or using it when in fact you shouldn't. Remember that Python requires indentation and for this it only accepts tab characters. Therefore, make sure that your text editor preserves tabs as there are many that silently replace them by spaces. The free editor Notepad++ is a good editor that does Python syntax highlighting and preserves tabs.

Besides indentation problems you may also run into Python compatibility issues. MicroPython is a port of Python, but it is not a perfect port. On the uPy website you can find a list of differences between CPython (the Python reference implementation) and uPy. So, if one day you run into a strange error where code that is supposed to work, doesn't, maybe you should check this list.

Setting up a simple webserver

Since the Kickstarter campaign had promised WIZ810io network support and since I happened to have a few of these modules lying around, I decided to try to set up a simple webserver (**Figure 4**). This was the point where things became complicated and clouds started to gather. The first step was, of course, Google in the hope to find an all-singing, all-dancing example, but I came up empty-handed. I did find some MicroPython webserver code [4], but it didn't work on my pyboard. Now, before you start

shouting and making signs to attract my attention, yes, I did reprogram the board with a special firmware that includes network drivers ([pybv11-network-20170322-v1.8.7-461-g58f23de.dfu](#)).

I had pasted the simple-enough web-server code in `main.py` but it didn't work. How to go about to debug it? Actually, I don't know what the preferred Python way is. I guess, either you add debug statements to the program or you can paste the program line-by-line into the REPL, which is what I ended up doing. Albeit somewhat tedious, this technique quickly showed me that the call of the `sendall` method of my `socket` object was invalid (read the code [4] to understand what I am on about). There is no such method! Now this is strange, because the MicroPython socket documentation clearly mentions it on its website. This took me to the MicroPython source code repository on GitHub to have a closer look at the network driver where I discovered, after a lot browsing, that the WIZnet driver does not export a `sendall` method at all. It doesn't export some

other standard sockets methods that work in MicroPython either and some of which even must be used. Go figure. Somebody didn't test things all the way through at MicroPython Corp. Inc. Ltd. OK, the W5200-based WIZ810io module is not the most recent one.

Anyway, even though replacing `sendall` by `send` was easy enough, it did not make my webserver work. It took another good deal of head scratching to find the culprit: a missing `content-length` field in the HTTP page header sent by my webserver. After adding it, my MicroPython webpage finally showed up in my browser (**Listing 1**).

Putting the working code in the file `main.py` lead to new surprises. Initially it worked fine, but when I started improving it a bit, something went horribly wrong and it all fell over. To cut a long story short, the file system on my pyboard had somehow become corrupted and the webpage's contents had been transformed into garbage. Also, the file `main.py` had become inaccessible and its size was reduced to 0. After

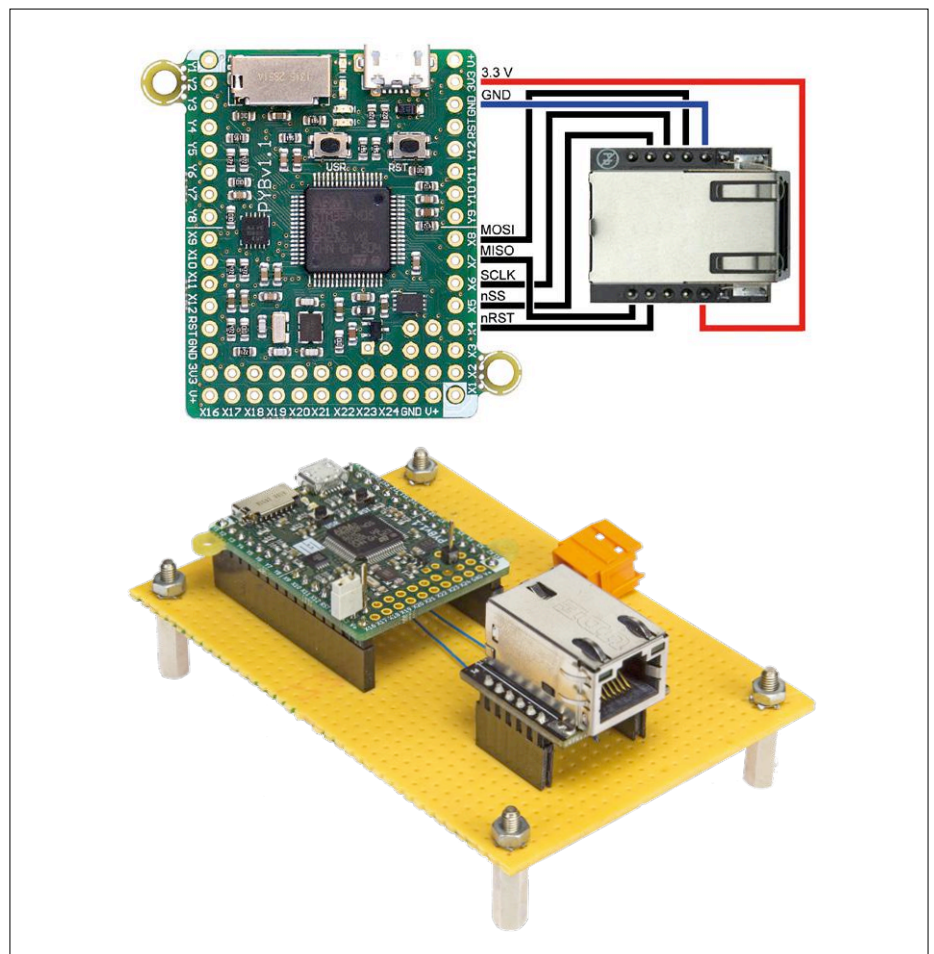


Figure 4. MicroPython webserver built with a PYBv1.1 pyboard and a WIZ810io Ethernet module from WIZnet.

restoring the pyboard using the methods described above, I could reinstall my little program to get the webserver up and running again.

This experience taught me to do the experimentation in the REPL instead of saving a file too often on the pyboard. Its filesystem seems to be a tad fragile. For comfortable REPL copy-pasting it is useful to have a serial terminal that supports multi-line pasting. Tera Term does this. Such a tool allows you to paste blocks of code up to points of where a manual backspace is required to decrease the indent level (for instance at the end of a `while` loop or `if-else` statement), speeding things up a bit.

I want more!

The MicroPython website has nice tutorials that show how to use the pyboard's peripherals. Reading them is highly recommended and will save you a lot of time. They also present some advanced techniques like combining assembler with uPy. Keep in mind that these tutorials concern the pyboard and presented techniques may not work (in the same way) on another board.

Conclusion

MicroPython and the pyboard present a fun and low-cost way of getting into embedded programming and learning Python along the way. All of the goods promised during the Kickstarter campaign have been delivered, even the stretch goals. The uPy website with its nice tutorials may not always be up to date, so, when in doubt, consult the project pages at GitHub. ◀

(160428)

Web Links

- [1] www.elektormagazine.com/160428
- [2] <http://micropython.org/>
- [3] <https://github.com/micropython>
- [4] <https://github.com/RinusW/WiPy/tree/master/AiCWebserver>



FROM THE STORE

→ SKU17931:
MicroPython pyboard v1.1

→ SKU18023:
Python for microcontrollers:
getting started with MicroPython

Listing 1. Simple webserver for the pyboard.

```
import os
import network
import socket

# Adapt these for your network.
my_ip = '192.168.2.33'
subnet_mask = '255.255.255.0'
gateway = '192.168.2.1'
dns = '8.8.8.8' # dunno what to put here, 8.8.8.8 is WIZnet example.
# The one and only page we serve.
page_name = 'page1.htm'

# Initialize network interface.
nic = network.WIZNET5K(pyb.SPI(1), pyb.Pin.board.X5, pyb.Pin.board.X4)
nic.ifconfig((my_ip, subnet_mask, gateway, dns))
print(nic.ifconfig())
# Launch server (2x Ctrl-C gets you back into REPL mode).
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(0)
while True:
    conn, addr = s.accept() # addr is not used but required.
    request = conn.recv(1024) # receive HTTP request.
    # Look for the string "Val=".
    val_begin = str(request).find('Val=')
    if val_begin > 0:
        # Found string "Val=", now extract value.
        pyb.LED(1).on()
        val_end = str(request).find(' ', val_begin)
        v = str(request)[val_begin+4:val_end]
        print("Val =", v)
        conn.send(v)
        # LED3 off if v < 50, on otherwise.
        if int(v) < 50:
            pyb.LED(3).off()
        else:
            pyb.LED(3).on()
    else:
        # String "Val=" not found, serve web page.
        pyb.LED(2).on()
        # Send HTTP header.
        conn.send('HTTP/1.1 200 OK\nConnection: close\nServer: pyboard\nContent-Type: text/html\n')
        conn.send('Content-Length: ')
        # We have to insert the size of the data we are going to send.
        conn.send(str(os.stat(page_name)[6]))
        # Unreadable way of getting file size.
        # Close HTTP header.
        conn.send('\n\n')
        # Send web page.
        f = open(page_name, 'r')
        conn.send(f.read())
    conn.close()
    pyb.LED(1).off()
    pyb.LED(2).off()
```




(almost) everything
you wanted to know about...

Components in Space

Interview with **Jaime Estela**, Spectrum Aerospace Group

Now that small companies and even start-ups can afford their own satellites, it's good to know more about components in space.

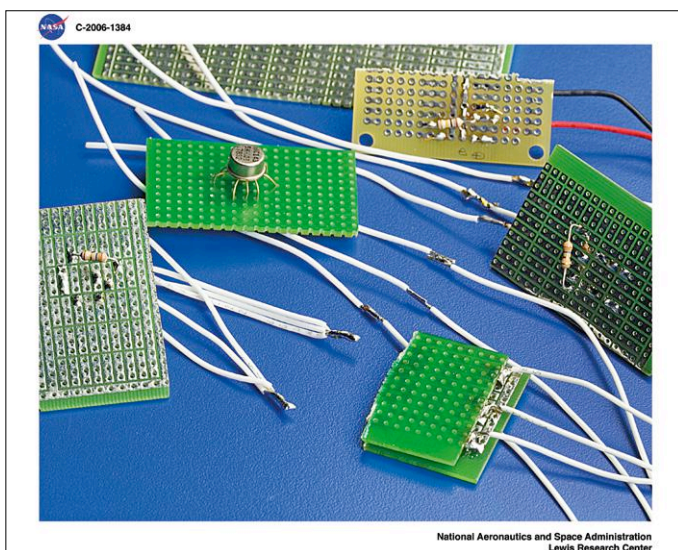


Figure 1. ISS astronauts practicing repair tasks for electronic components (photo: NASA).

Q Are electronic components in space degraded by radiation?

A All electronic components in space are degraded by radiation. The robustness of components depends on many factors. Some components only have a short life in space, while others can operate in space for many years. Components specifically designed for military use can function in space for a very long time. Commercial parts vary. Even if they are not made for use in space, many components can function in space for several years with only minor degradation.

Q What are the effects of radiation in space?

A **Displacement damage (displacements in the crystal structure):** Particles (neutrons, protons, alpha particles and heavy ions) and high-energy gamma photons alter the structure of the crystal lattice and affect the properties of semiconductor devices.

Accelerated aging (ionizing radiation): Charges are induced and accumulated. These charges increase the leak-

age currents of the components, causing higher power dissipation which ultimately causes failure of the components.

Single event effect: Highly ionized particles induce strong transient discharges (noise pulses) in semiconductor devices, which interfere with device behavior and cause data corruption.

Q How are components tested for radiation resistance?

A There are two main types of radiation test in practice:

Total ionizing dose (TID) test: For this test a cobalt-60 cell is used as a gamma source. Similar sources are used in medical equipment, but there they are much weaker. The radiation exposure simulates several years of deployment in space in a very short time. The radiation-induced charges affect the properties of semiconductor devices in various ways.

Single event effect (SEE) test: In this test a particle accelerator is used to bombard the component with protons or heavy ions. The most serious result is data corruption due to short noise pulses induced in the circuit. This effect is highly relevant for digital circuits.

Q Are there differences with regard to the satellite orbit?

A Yes – the higher the orbit, the higher the radiation dose. Low earth orbit (LEO) satellites are exposed to the least radiation doses. Passengers in aircraft flying at an altitude of 10 kilometers (33,000 feet) also receive higher radiation doses, and the active flying hours of crews are limited to avoid endangering their health. The ISS space station orbits at a height of 350 to 450 km. This orbit was chosen to minimize the radiation dose received by the astronauts.

The Van Allen Belt provides additional protection. Objects inside the Van Allen Belt are protected by its magnetic field. The highly charged particles are captured by the magnetic field and diverted to the polar regions.

Telecommunications satellites are positioned in geostationary Earth orbit (GEO). Satellites in this orbit appear to hover over the same spot on the surface of the Earth. The height of the geostationary orbit is nearly 36,000 km, putting it well outside the Van Allen Belt. This orbit is more demanding on electronic

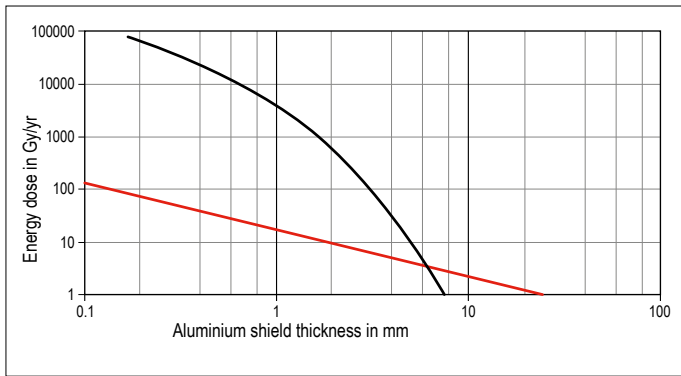


Figure 2. Radiation protection by aluminum shielding in geostationary orbit (black curve: energy dose due to electrons) (source: Wikipedia, public domain).

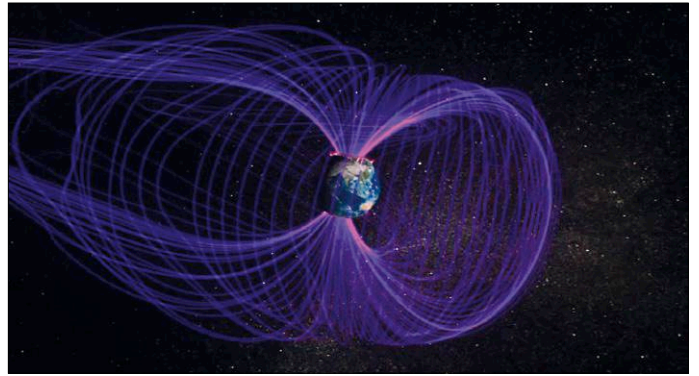


Figure 3. The first US satellite mission in 1958, Explorer 1, discovered a belt of electrically charged particles surrounding the Earth (graphic: NASA).

devices because they must operate under higher radiation levels. Deep space missions are even more difficult. Most of the electronic devices are located inside the protective structure of the satellite.

Q *What types of components are used in space missions?*

A The longer and higher the mission, the more protective measures are necessary. The design must fulfill the requirements – no less, but also not too much more. Components classified as “International Traffic in Arms Regulations” (**ITAR**) can easily function for 100 years in LEO, but if the planned mission lifetime is relatively short – for example, only five years – these components are unnecessarily expensive. For this reason, commercial off-the-shelf (**COTS**) components are being used more and more.

Geostationary satellites need to be able to operate reliably for 15 years in a distinctly harsher environment. Using ITAR components is therefore worthwhile. It often happens that the electronics of these satellites still function reliably even after the end of their planned lifetime.

Another situation is high-altitude research rockets, which are airborne for only a short while (at most 15 minutes). Here there is no need for special components; COTS components are fully sufficient.

Q *Which protection mechanisms are the most effective or most reasonable in various situations?*

A Almost all satellites are built with **redundancy**. Pico satellites and small nano satellites, which do not have enough room for redundant hardware, form an exception. Redundancy helps to increase the useful operating time. In the early days of space flight, all systems had multiple redundant modules. For example, up to eightfold redundancy was employed for the on-board computer. Especially for manned missions, the electronics must work flawlessly. Although redundancy increases the complexity, power consumption, weight and size of the space vehicle, it is still one of the most important ways to increase the reliability of aerospace systems.

Fault tolerance: It is always advisable to design the electronics with robust components. You can even take the known degradation of the components into consideration in

the design, so that the electronic functions are always inside the operational envelope. On the data sheet of a commercial component you cannot see how it will behave under the conditions of a space mission. Even the manufacturer does not know. However, these properties are very important for deployment in space, so they must be determined empirically.

Shielding: It is common practice to protect electronic devices by placing them inside the satellite structure. Increasing the thickness of the shielding improves the protection, but it also increases the size of the structure and, worse yet, the weight of the satellite. These two factors raise the launch costs. It is also possible to use other materials, but most of them are heavier than aluminum. For this reason, special shielding is only used where it is absolutely necessary.

Q *Which components are especially vulnerable, which are less vulnerable, and which are not vulnerable?*

A The key difference is between passive and active components. Resistors and other passive components are very robust, but semiconductors are strongly degraded by radiation. Bipolar technology is generally more resistant to radiation than CMOS technology. However, that depends on many factors, including the layer geometry, the materials, the packaging and so on.

Diodes, transistors, all ICs, crystals and optoelectronic components are sensitive to **ionizing radiation**, while FETs, CCDs, CMOS active pixel sensors, other optoelectronic components, and all ICs are sensitive to **single event effects**.

ESA and NASA now promote the use of FPGAs because programmable hardware can be repaired remotely. In addition, FPGAs pack a lot of functions in a single component, making the electronics more compact. ESA also has a processor called LEON, which is a soft IP core that can be implemented on an FPGA [1].

Q *What are the effects of degradation or damage?*

A One effect of degradation is that the electronics can consume more power. Progressive degradation ultimately leads to functional failure of components.

If you know how a component behaves in space, you can take this into account in the design. For example, if you have an amplifier with a gain of 5 and you know from simulation or a qualification test that the gain will drop to 3 by the end of the mission, you can incorporate this information into the circuit design so that the circuit will function properly over the entire mission lifetime. That way you can counter the degradation of the component.

If you use components in aerospace electronic designs without knowing how they behave in space, you are taking unnecessary risks. The most professional approach, but also the most expensive, is qualification of individual components. Nowadays there are satellite operators who want to save time and money by performing very short tests on fully assembled boards. If a board fails the test, you have no idea which component actually failed. The robustness of the board is only as good as the weakest component. If on the other hand you know the behavior of every component, you are in a good position to make the board more robust.

A TID test takes about two weeks and follows ESA recommendations. Sometimes people want to have a test performed in just six hours. That is like baking a cake: if the package says "bake 30 minutes at 200°C" and you instead decide to bake it for only 10 minutes in an oven at 600°C, you will not obtain the same result. You cannot fool physical processes. ◀

(160299-1)

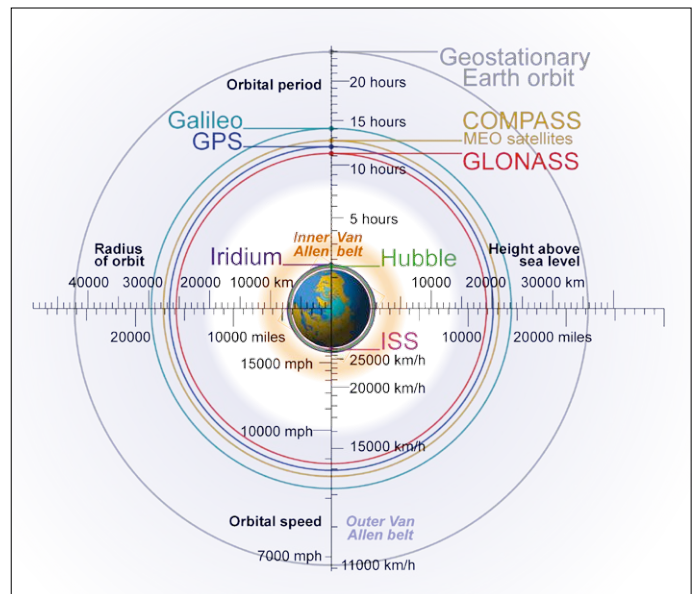


Figure 4. Orbital positions of satellites and the ISS in the Van Allen Belt as seen from the North Pole (graphic: Geo Swan, CC BY-SA 3.0).

References

Electronics for Space Guide,
by Jaime Estela, available soon from Elektor.

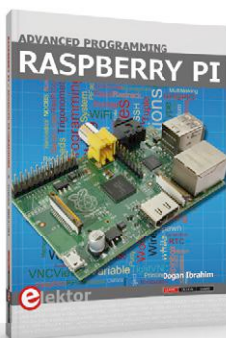
Advertisement

LEARN DESIGN SHARE

Raspberry Pi @ Elektor

It's very easy to connect the Raspberry Pi 3 to a computer or a television set. You can program this high-performance computer yourself, use it to play videos, play games with it, and use it to develop your own highly innovative concepts.

www.elektor.com/rpi3



Raspberry Pi Advanced Programming
www.elektor.com/rpi-book



Raspberry Pi 3 Starter Kit (Deluxe)
www.elektor.com/rpi-deluxe



pi-top DIY Laptop Kit
www.elektor.com/pi-top



More Raspberry Pi at www.elektor.com/rpi

The ESP8266's Big Brother

Getting started with ESP32 and the Arduino IDE

The ESP8266 from Espressif has made waves in the maker community with its low price: you can get one from AliExpress for just two or three euros/dollars/pounds. Its big brother, the ESP32, is the result of Espressif's efforts to make their WLAN SoCs fit for the future. Along with a dual-core processor, it has onboard Bluetooth, more RAM, and IO extensions. In this article we talk about programming the ESP32 with the Arduino IDE. In the next issue of *Elektor* we will turn our attention to the native development tool IDF.



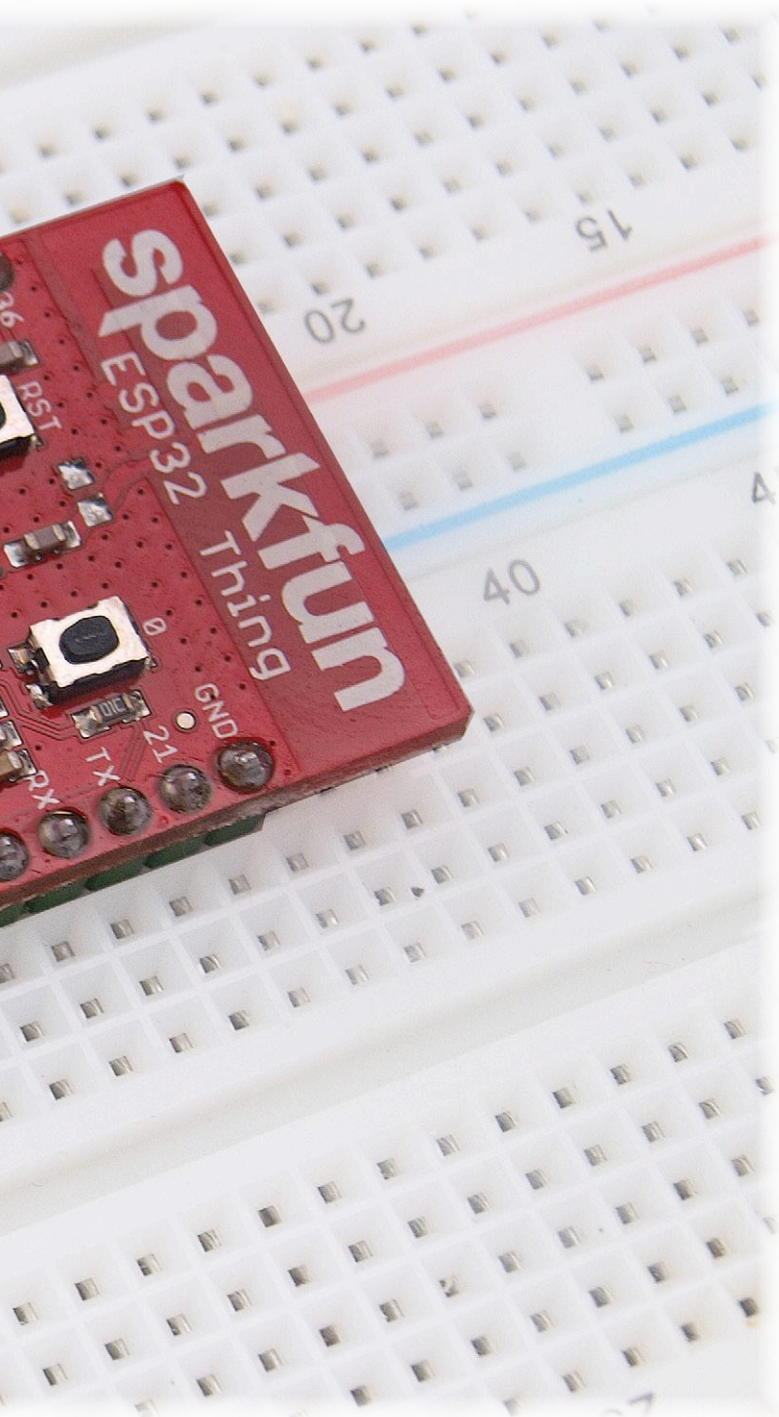
By **Tam Hanna** (Slovakia)

After the initial delivery of the ESP32 there were massive supply problems with the breakout boards, although individual ICs were only slightly affected. Due to the very small pin pitch, developing your own PCBs for prototype testing is not an attractive option. For this article we are using the SparkFun ESP32 Thing, which can be obtained from Mouser [1] and other suppliers. The SparkFun board does not come with any pinheaders; the pins necessary for connection to a breadboard must be purchased separately. Along with the ESP32 Thing, there are now

dozens of other boards available from various manufacturers. Espressif maintains a list of suitable candidates at [2], which largely behave the same way as long as they have a USB IC from FTDI.

A workstation for the ESP32

The ESP32 Thing is equipped with the familiar USB to serial converter IC from FTDI. That is advantageous for users who, as owners of an Arduino board or something similar, already have the corresponding driver on their PC. The following description is based on the Windows 8.1 Enterprise operating system and version 1.8.2 of the Arduino IDE.



Annoying bugs

You should be aware that early versions of the ESP32 had a number of bugs. A useful list (recommended reading) can be found at [9].

which holds the SDKs manually entered by the user, but there is nothing to worry about if this folder is missing.

Now you can determine what you should enter in the *Target Directory* field — on the author's PC this is `C:\Users\TAMHAN\Documents\Arduino\hardware\espressif\esp32`.

Then click on the *Clone* button to start downloading the code from the GitHub repository. This may take a while, depending on the current load on the download server. You should not worry if the Git window sometimes does not respond. After the download is completed, the multicolor window shown in **Figure 1** appears. You can close it unchanged.

Remarkably enough, the SDK is supplied in two blocks. From GitHub you only receive the data that is specifically related to the Arduino environment. To download the data related to the general ESP32 components, you need to use the program `C:\Users\TAMHAN\Documents\Arduino\hardware\espressif\esp32\tools\get.exe`, which with a bit of luck you can run by double-clicking in Windows Explorer. This program opens a console window which shows the download progress. After the download is completed, you will have an additional directory named `C:\Users\TAMHAN\Documents\Arduino\hardware\espressif\esp32\tools\xtensa-esp32-elf`, which contains the ESP IDF development environment that will be discussed in the next article of this series.

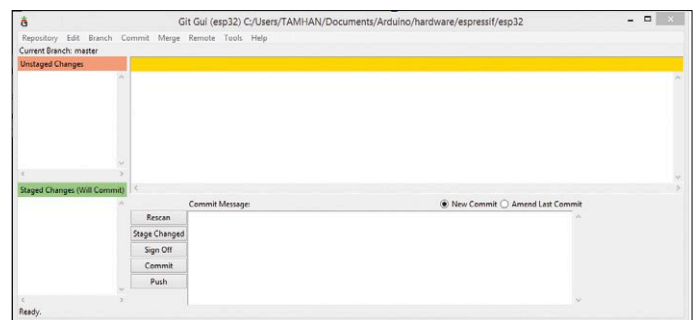


Figure 1. Downloading the Arduino SDK for the ESP32 from GitHub.

At the time of printing of this article, Espressif had not yet installed the SDK (“core”) in the board manager of the Arduino IDE. In order to download the code you need the graphic front end for Git, which is available from [3] and nestles under “Git GUI” on the Start menu. Click on Clone Existing Repository and then enter `https://github.com/espressif/arduino-esp32.git` in the *Source Location* field.

The higher-level target directory is `C:\Users\TAMHAN\Documents\Arduino`. Here you must replace “TAMHAN” by the user name you employ for running the Arduino IDE on your computer. Depending on the configuration, this directory may contain one or more sketches and possibly a folder named “hardware”

Next, connect the ESP32 Thing to your PC with a Micro USB cable to start the process of searching for the driver. Don't be surprised when the wizard shows a mouse symbol for this. For reasons beyond the author's understanding, the FTDI driver is outfitted with this icon. After the device has been recognized, you can click on *Tools* → *Board* and select *SparkFun ESP32 Thing*. On a relatively small monitor you may have to scroll down to see this entry, since the Arduino IDE always shows the boards in the Arduino family above any third-party boards in the board selection list.

The Arduino IDE does not automatically recognize the ESP32 Thing. To determine the right COM port, open the Device Manager window, where you will see the device listed as *USB Serial*

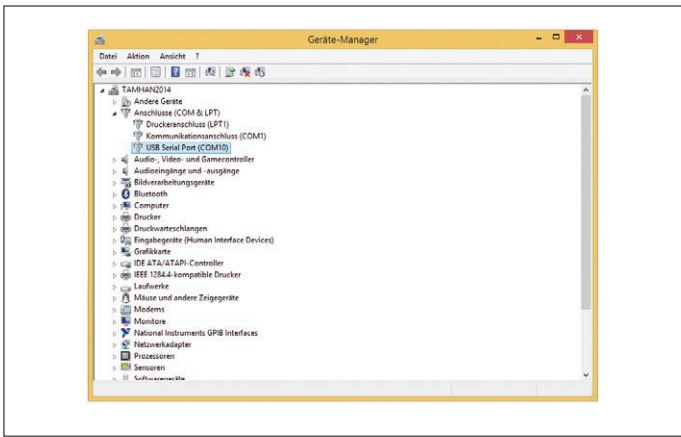


Figure 2. You can see the COM port number in the Device Manager window.

Port as shown in **Figure 2**. Enter this port ID in the Arduino IDE to complete the configuration. The default upload speed of 921,600 bps can be left unchanged for now. If you experience problems with transferring binary files, especially very large ones, you should follow the lead of experienced ESP32 developers and reduce the data transmission rate. That naturally leads to longer cycle times for reprogramming, so you should avoid being over-cautious in this regard.

Hello World!

Designers of evaluation boards normally equip their products with one or more LEDs, and pin 13 has become the preferred choice for this in the Arduino world. True to their nature, SparkFun does not follow convention and puts the LED on pin 5. The sketch shown in **Listing 1**, which is included in software for this article which you can download from the Elektor site [4], is therefore a good choice for the first test.

There is nothing remarkable here from a programming perspective — the LED is periodically switched on and off. After the deployment process, which takes 30 to 60 seconds, the blue LED starts blinking at regular intervals.

How fast can it go?

The next thing we want to find out is how long it takes for the ESP32 to complete processing of commands, which is import-

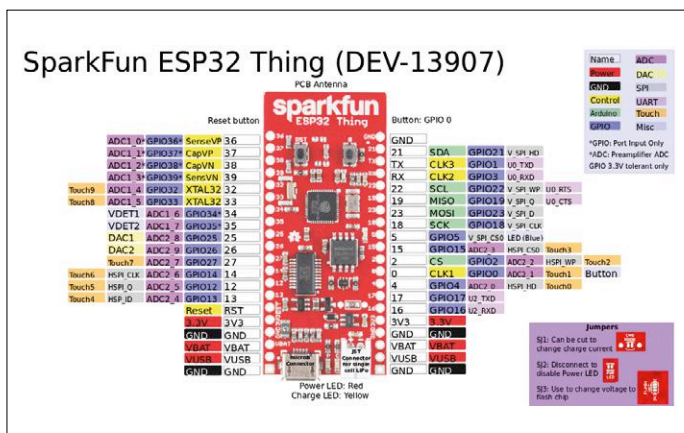


Figure 3. The pinout of the ESP32 Thing is fairly complicated.

ant information in an era of real-time operating systems. The Arduino 101, for example, gives itself up to 8 seconds to think about the command, which in some cases can lead to hazardous operating situations in instrumentation and control applications. In order to identify the appropriate pins for making measurements, we first have to come to grips with the hardware architecture. In the ESP IDF you can find the structure `gpio_num_t`, which is responsible for addressing the GPIO ports. This code (shown here highly abridged in the interest of better readability) takes the following form:

```
typedef enum {
    GPIO_NUM_0 = 0,      /*!< GPIO00, input and output */
    GPIO_NUM_1 = 1,      /*!< GPIO01, input and output */
    . . .
    GPIO_NUM_39 = 39,    /*!< GPIO39, input mode only */
    GPIO_NUM_MAX = 40,
} gpio_num_t;
```

From the linear layout of this structure you can conclude that the ESP32 does not work with GPIO pin banks. Instead, each pin is uniquely described by its own ID. Closer examination of the code is not necessary here. As can be seen from the following method, you can use the supplied IDs directly in your own code:

```
extern void IRAM_ATTR __digitalWrite(uint8_t pin,
uint8_t val) {
    if(val) {
        if(pin < 32) {
            GPIO.out_wlts = ((uint32_t)1 << pin);
        }
    }
    . . .
```

Armed with this knowledge, you can consult the data sheet of the ESP32 Thing, which can be downloaded from [5] as a PDF file and contains the graphic shown in **Figure 3**. If you are using a different development board, you should consult the documentation from the corresponding manufacturer.

Here we are interested in the two pins GPIO5 and VUSB. The first is connected to the LED which is switched by our program, and the second is connected directly to the Micro USB port and tells us whether or not voltage is present on that port. Since the LeCroy 9354AM oscilloscope we used for the following steps has four channels, we decided to also monitor the 3V3 pin, which is connected to the output of the voltage regulator on the PCB.

Developers with a background in the Raspberry Pi domain need to be very careful here, because the ESP32 is a pure 3.3 V microcontroller and will be permanently damaged by applying voltages above 3.6 V. Unlike the Upton processor, the ESP32 is absolutely not 5 V tolerant. This is also an issue when the board is connected to bus systems such as SPI or I²C.

The current specs call for a minimum supply voltage of 2.2 V, which means that if you mount the ESP32 on your own board you do not necessarily need to have a 3.3 V supply voltage available. Incidentally, the voltage range of 2.2 to 3.6 V looks a lot like the normal voltage range of lithium ion cells, which is most likely an intentional application scenario.

However, let's get back to our main task now. An analysis of the oscilloscope output shows that the ESP32 needs about

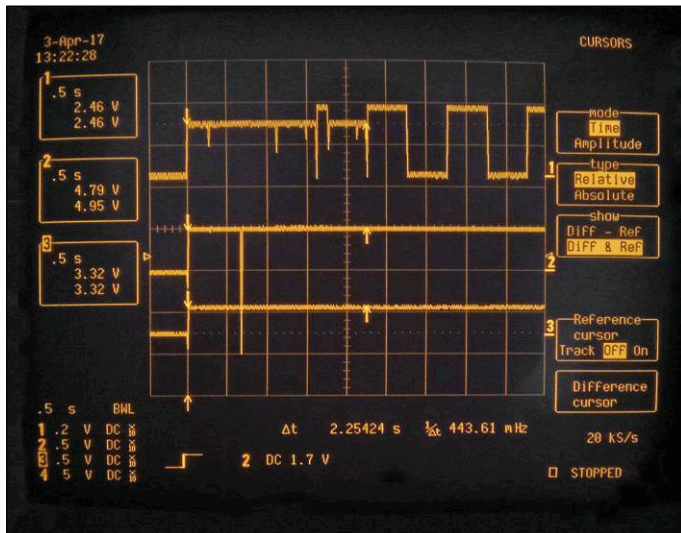


Figure 4. It takes more than two seconds for the blink program to start running.

2.5 seconds to get ready for action (see **Figure 4**).

In this connection it should be noted that the evaluation board uses a relatively complex boot loader. During this processing time the LED is only halfway lit, as can be seen from the first channel of the oscilloscope.

Something special

At this point we were tempted to pull out our modulation domain analyzer and measure how fast the outputs can be switched, but then we had second thoughts because it is unlikely that the ESP32 would ever be used for hard real-time tasks, due to the use of FreeRTOS as well as the relatively large effort involved. A much more attractive prospect is to see what can be done with the many digital to analog converter options on the ESP32. Arduino users commonly employ the method `analogWrite` for this, but unfortunately it is not yet supported by the Espressif SDK (at the time of printing of this article). The manufacturer recommends three alternatives for this. The first is the PWM processor designated as LEDC, which can process up to 16 pins at the same time [6]. The second is the sigma-delta module [7], which can handle up to eight channels. The third is the conventional DAC [8], which is limited to just two channels. Here we opted for the third method, with the objective of implementing a simple function generator – always a popular task. If instead you want to try one of the other methods, you can follow one of the indicated links. They lead to header files which describe the modules concerned.

Working with the DAC

As always, it is a good idea to first look at the implementation of the API. Open the appropriate link in the browser of your choice, with the suffix `.h` changed to `.c`. For the DAC the relevant method is called `__dacWrite`, and its body looks like this:

```
void IRAM_ATTR __dacWrite(uint8_t pin, uint8_t value) {
    if(pin < 25 || pin > 26){
        return;//not dac pin
    }
}
```

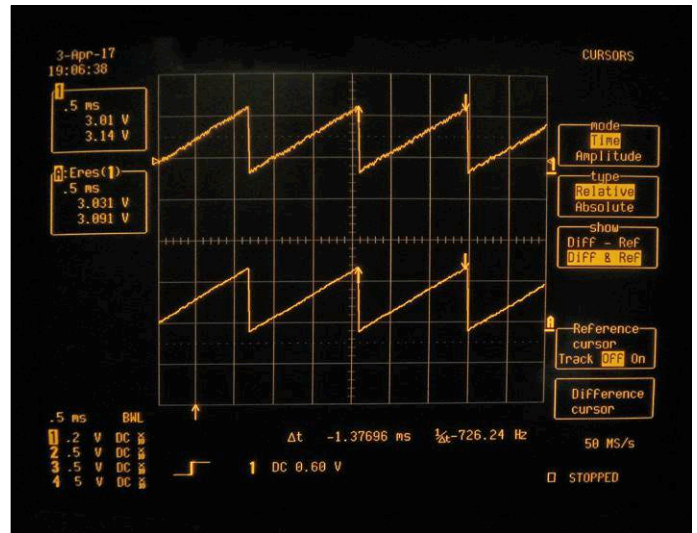


Figure 5. You can use the DAC to output user-defined signals.

```
pinMode(pin, ANALOG);
uint8_t channel = pin - 25;
```

There are two noteworthy points here. Firstly, it has become common practice in the ESP32 community to identify the two channels by the numbers of their associated GPIO pins. The method therefore expects to receive the value 25 or 26, which it first converts into a channel ID.

Remarkably enough, it appears that Espressif does not trust users of this method to manage the state effectively. For that reason the method `pinMode` is activated again for each call, which obviously does not enhance efficiency.

Now we can get back to the task of implementing waveform output. The DAC is an 8-bit device, so its input variable has a value range of 0 to 255:

```
void loop() {
    for(int i=0;i<255;i++) {
```

Listing 1. Our first program for the ESP32

```
//File ElektorES032Scratch1

int ledPin = 5;

void setup()
{
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
    delay(500);
}
```

```

    dacWrite(25, i);
  }
}

```

Now we can connect an oscilloscope to the selected pin and check the waveform of the output signal, which is shown in **Figure 5**. The duplication of the trace here is due to a special feature of the LeCroy oscilloscope – along with the original trace, it shows a second trace with the resolution increased by 3 bits.

WLAN connection

The SDK provided by Espressif comes with a set of demo programs which illustrate the use of the network API: They are located under *File → Examples → Examples for SparkFun ESP32 Thing*.

Here we decided to check out communication with the Internet. Since there are a few hurdles to be overcome in that process, we implemented the program in two stages. Our first demo is intended to show that we can establish a connection to the WLAN.

First we need the name of the WLAN in the form of a string: This parameter is case sensitive, so if possible you should use copy and paste to enter it in the program. A good way to see the name of your WLAN if you are working on a Windows platform is to use the *netsh* command – but don't be surprised if only part of the response fits in the terminal window. On the author's computer, the relevant block looks like this:

```

C:\Users\TAMHAN>netsh wlan show all

. . .

===== SHOW INTERFACES =====
There is one interface present on the system:

. . .

        SSID                : Tamoggemon Holding k.s. BAa

```

The WLAN header `<WiFi.h>` must be included in the program code, and two other variables must be declared to hold the name and password of the network concerned:

```

#include <WiFi.h>

const char* ssid    = "Tamoggemon Holding k.s. BAa";
const char* password = "XXX";

```

The next step is to perform the housekeeping tasks. Along with setting up the serial connection to the PC, we introduce a delay to give the ESP32 enough time to process the configuration commands:

```

void setup() {
  Serial.begin(115200);
  delay(10);
}

```

Now we start the actual process of establishing the connection. In this regard it is relevant that `WiFi.begin(...)` returns immediately. However, we can check for a successful connection by polling `WiFi.status()`:

```

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}

```

Last but not least, we output the IP address that was assigned to the router by the WLAN infrastructure:

```

Serial.println("");
Serial.println("connected - IP is:");
Serial.println(WiFi.localIP());
}

```

The body of the loop method can be left empty for now. Now we run the program with the serial monitor window open to obtain the output shown in **Figure 6**. Note that we are using 115,200 bps here: When it is first opened, the serial monitor is set to a bit rate of 9600 and is therefore not able to decode the incoming data.

Data from the network

Now we can try downloading data from the network. We use the `WiFiClient` object for communication with TCP servers. `WiFiClient` does not care about protocols at the application level. For this reason, all it needs to establish a connection is the name of the target host and the port number to be used. It returns `false` if it fails to establish a connection:

```

void loop() {
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect("tamoggemon.com", httpPort)) {
    Serial.println("connection failed");
    return;
  }
}

```

In the absence of an HTTP driver, we have to formulate a valid request by hand: Since we are not particularly interested in the overall HTTP protocol for this article, you do not need to be concerned about the string shown in the listing:

```

String url="/test/2017/Elektorfile.txt";
client.print(String("GET ") + url + " HTTP/1.1\r\n"
+
    "Host: " + "tamoggemon.com" + "\r\n" +
    "Connection: close\r\n\r\n");
unsigned long timeout = millis();

```

Processing of HTTP requests may take a while, depending on the load on the server. Here we allow our server 5 seconds for this and trigger a timeout if we do not receive at least one byte within that interval:

```

while (client.available() == 0) {
  if (millis() - timeout > 5000) {
    client.stop();
    return;
  }
}
}

```

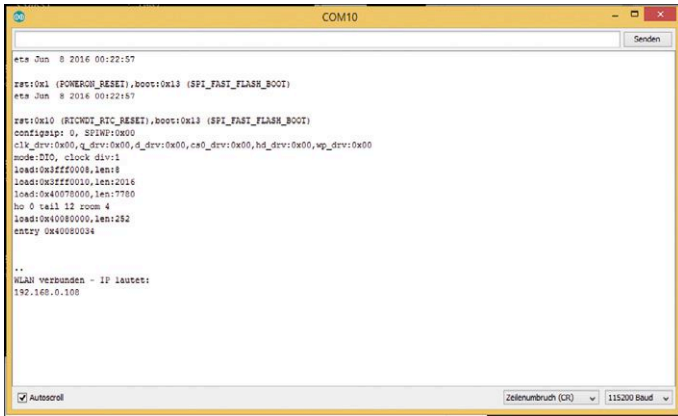



Figure 6. Now we're online!

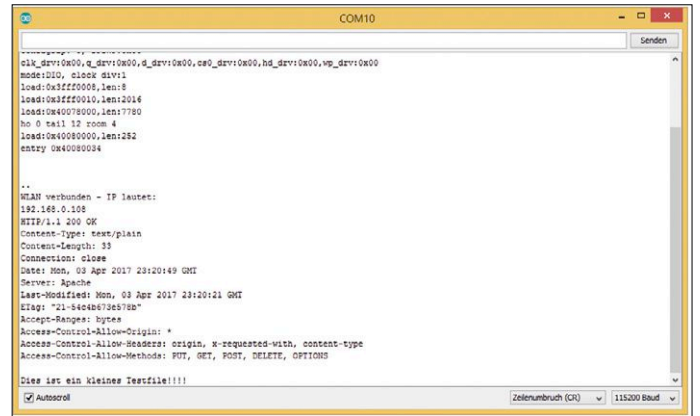


Figure 7. HTTP is very verbose.

After the data has been exchanged, we read it byte by byte from the client object and forward it to the serial monitor:

```
while(client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
}
```

To avoid creating an overload from constant polling of the server, we call the method `delay(...)`. It causes the ESP32 to take a 50-second break.

```
delay(50000);
}
```

Now we are ready to send the sketch to the ESP32. You shouldn't be alarmed when the console is filled with various outputs: That's because each HTTP request delivers an enormously long header in addition to the actually requested data (see **Figure 7**).

Summary

The Raspberry Pi Zero W and OrangePi Zero are serious competitors for the ESP32 because the prices of the breakout boards are roughly the same. However, if you are planning a small production run you can save a lot of money with the ESP32. For volume production, the extra effort of making your own PCB pays off because the individual ESP32 ICs are significantly less expensive than the breakout boards.

Our journey through the ESP32 world is not yet finished. In the next issue of *Elektor* we will devote our attention to the native API, which provides access to some additional functions and additionally enables higher performance. ◀

(160454-1)

Web Links

- [1] <http://eu.mouser.com/new/sparkfun/sparkfun-things/>
- [2] <http://esp32.net/#Hardware>
- [3] <https://git-scm.com/download/win>
- [4] www.elektormagazine.com/160454
- [5] <https://cdn.sparkfun.com/datasheets/Wireless/WiFi/ESP32ThingV1.pdf>
- [6] <https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-ledc.h>
- [7] <https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-sigmadelat.h>
- [8] <https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-dac.h>
- [9] http://espressif.com/sites/default/files/documentation/eco_and_workarounds_for_bugs_in_esp32_en.pdf

Electronics in Orbit

Space science for all of us

By **Jaime Estela** (Spectrum Aerospace Research)

Satellites are constructed from robust materials, simply because they have to perform flawlessly under extreme environmental conditions. Their electronics require the use of specially manufactured — and therefore extremely expensive — components. For small satellites (smallsats) regular commercial components offer a cost-effective alternative, although their behavior must first be qualified in simulation tests to international standards before deployment.



NASA astronaut Mike Hopkins removes a Dewar flask from the freezer to begin biological tests (photo: NASA).

In space there's a variety of physical phenomena that affect the performance of electronic components and materials. For satellite missions it's vital to know this environment extremely well in order to minimize any damage and/

or disruption to the electronics. The precise behavior of the components under such adverse conditions depends on the orbit of the satellite and can be simulated using software tools. The outcome of these simulations then assists satel-

lite developers to design their systems accordingly.

Figure 1 sets out the phenomena that determine the service life and dependability of electronic components:

- **Atomic oxygen:** UV light breaks up O₂ molecules into individual atoms of oxygen. This atomic oxygen is highly reactive and will consequently erode the outer surface of the satellite's structure. This 'space rust' affects the thermal behavior of the structure and the satellite. This is a very important issue, as it affects the satellite's thermal system.

- **Plasma:** Ionized gases produce electrostatic charges that stress the surface of the satellite electrically. Discharging these can interfere with the operation of the satellite and the instruments inside it.

- **Radiation:** A multitude of effects result from this phenomenon. Gamma rays degrade the electronic components. Protons and heavy ions can at best falsify digital data or at worst physically destroy the electronics aboard the satellite.

- **Micrometeorites and space junk:** The deadliest events in space are caused by small artificial or natural matter. The effects of micrometeorites or space junk should not be underestimated; they can damage or destroy satellites. This very thing has already happened, resulting in the loss of spacecraft.

Degradation in electronics

In the early days of space flight history, electronics designed especially for spacecraft did not exist. Instead electronic components designed for military purposes were upgraded for deployment in space. In this upscreening process various supplementary tests were conducted, with the best-performing components selected.

In 1973 military-grade components were again used in the hardware for the Skylab space station. In the qualification test the hardware had to be upgraded several times over on account of malfunctions that arose. These improvements resulted in the need for more than \$3 million of additional investment in procuring new electronic components required and on additional qualification exercises.

Electronics in the first Skylab mission (1981) once more employed military-specification components. In order to raise the dependability of the system, most elements were installed with six-

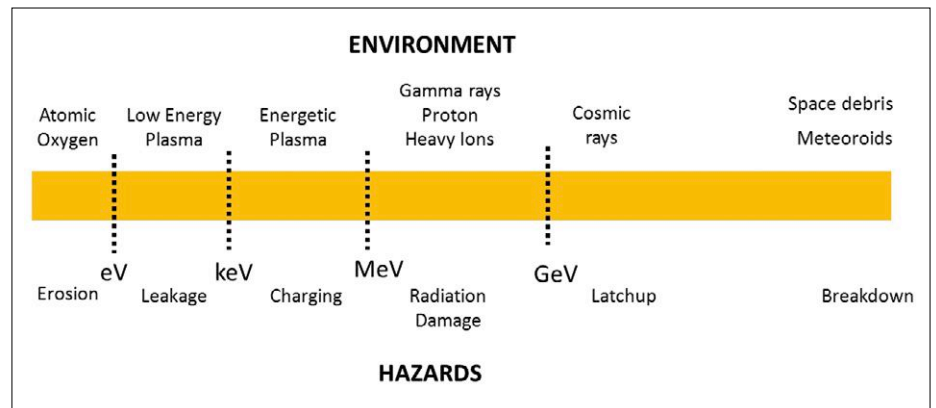


Figure 1. Environmental conditions in space and their potential hazards (source: Spectrum Aerospace)

fold redundancy. Valid data was checked using a statistical method (voting algorithm). However, this inclusion of redundant systems also enforced an increase in weight and power consumption as well as other challenges to hardware and software design.

As military-grade electronics did not really lend themselves to space flight applications and the upscreening produced no improvement of the components, work began in the 1960s on the systematic development of dedicated space electronics, which demanded qualitatively high-end production techniques. As a basis for this new generation of components in the USA military components were selected, with complementary tests defined for their qualification. This strategy made possible a reduction in production costs for space flight missions, at the same time achieving a high level of ruggedness in the components. However, because the demand for components of this kind was obviously very small, their price remained high.

Nowadays the qualification process for just one component to European Space Agency (ESA) or U.S. National Aeronautics and Space Administration (NASA) standards requires an investment of a million dollars or more. The time taken for certification is around two years [1]. Of course the entire test process can turn out to be complex or simple, according to the type of component. By way of example, testing a diode is less cost-intensive and protracted than checking out a microcontroller.

Qualification tests

The experience gained over the last sixty years during numerous space missions and using various technologies enabled

institutions like ESA and NASA to devise guidelines for developing qualifications and standards. These guidelines underpin the professional qualification of electronic components for space flight missions. The standards laid down by ESA can be found on the ESCIES (European Space Components Information Exchange System) portal [2]. A database of qualified integrated circuits together with their test results is available on the ESCIES website [3].

For a qualification test the components under examination (Devices Under Test, DUTs) are made ready and the measurement environment configured. First the precise number of components is determined and depending on the type of test, either a small number or the entire quantity of the components is nominated. This means that sometimes just a few components will suffice for the qualification (for example during radiation testing), whereas for an outgassing test (for instance) all of the components must be tested. During testing the DUTs are characterized.

An important requirement in qualifying components is that all of the components for qualification must be identical (in geometry, size, materials and so on) and must also come from the same production batch.

The test results are collated in a test report. Following qualification a test report will describe the behavior of the relevant component parameters under space flight conditions. A complete, highly extensive qualification is known as a 'screening test'. A process of this kind takes a couple of years and requires a high level of financial investment. In the screening test the following individ-

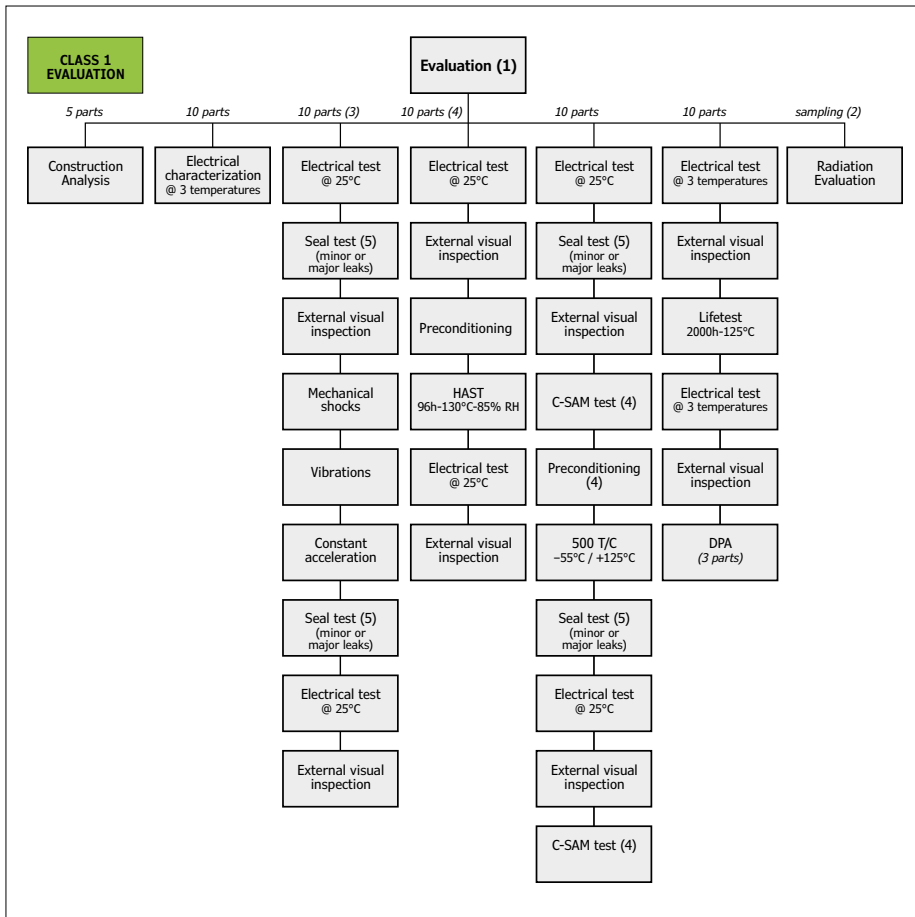


Figure 2. Evaluations tests carried out in Class 1 (ECSS-Q-ST-60-13C Evaluation tests flow chart Class 1, source: ESA).

ual tests are undertaken:

- Electrical test
- Impermeability testing
- Visual inspection
- Mechanical shock
- Vibration test
- Constant acceleration
- Thermal test
- Radiation test
- Outgassing
- Stress test
- Thermal shock
- Solderability

ESA document *ECSS-Q-ST-60-13C Space Product Assurance* describes the various quality classes [4]. The differences between these classes lie in the depths of the qualification processes. Class 1 is concerned with complete qualification (**Figure 2**), whereas Class 3 requires only a short qualification process (mainly radiation tests).

Characterization

The way in which these qualification pro-

cesses are carried out can be seen in a brief description of the TID (total ionizing dose) testing of an LTC2052 drift-free operational amplifier. Firstly, a rack is prepared which contains circuit boards with the components to be inspected (**Figure 3**). In this case, the two configurations (bias condition) shown in **Figure 4** are of importance: an 'off mode' in which all pins are connected to system ground and an 'on mode', in which the device is configured for a specific operating point but the component is not processing a signal (that is, in a kind of standby state).

Figure 5 shows the test setup. Cobalt-60 is used as the radiation source. The two test configurations are subjected to irradiation. During irradiation, the test specimens are measured from time to time by means of automatic test equipment (ATE). These characterization results show the degradation of the components at ever-increasing levels of radiation dose. The reduction in performance depends on the technology of the device and the accumulated radiation dose. For

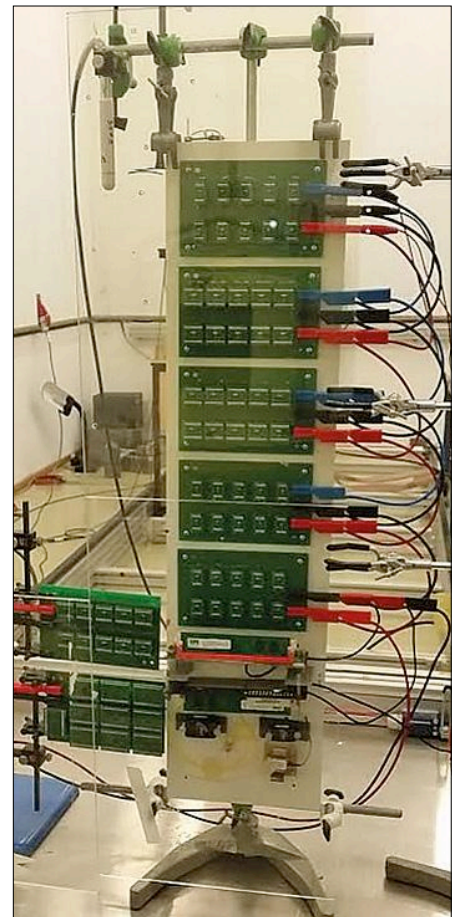


Figure 3. Array of devices under test ready for radiation testing (source: Spectrum Aerospace).

example, rising current consumption can be observed with increasing radiation dosage (**Figure 6**). At the end of the test changes in electrical parameters are detected; in some cases components have even been completely destroyed.

The smallsat marketplace

The market for small satellites is showing fairly rapid growth. New products and functionalities are sought to match the growing requirement. For manufacturing smallsat hardware military ITAR (International Traffic in Arms Regulations) components are not the right solution; qualified COTS (Commercial Off-The-Shelf mass-produced products) are. Market research carried out by analysts Northern Sky Research indicates clearly that in the next ten years more than 2,500 small satellites (weighing up to 100 kg) will be launched into space [5]. Another market study, by the firm Spacework, forecasts 3,000 smallsats up to 50 kg in the period 2014 to 2020 [6]. Both forecasts envisage future constellations of satellites (effectively mass-pro-

Q & A, by Jaime Estela

- A competition was planned to win a place for a circuit board on board a StarLab flight. What has become of it?

Yes, this is still live. However, delays have arisen because a firm that wanted to co-sponsor the prize competition and had experience in events of this kind has dropped out. We are in discussion with other firms but this will definitely take a bit longer.

- When will the first flight take place?

The first flight should take place in December, although while some uncertainties remain, we don't yet have a fixed date. In principle flights should then be made in a three-month cycle.

Can you give a ballpark idea of your charges?

The price depends on the size of the PCB. For a small board (5x3 cm²) you can reckon on around €3000 / £2500 / \$3260 upwards.

- How will the experiments be monitored?

Electronics in space are observed using telemetry data. Sensors measure specific parameters and the test results are then transmitted to Earth. This way you always know exactly how your space electronics are functioning. Developers need to concentrate on their own experiments nevertheless, for which reason a StarLab development kit is provided. This electronic support package assists developers to prepare experiments involving telemetry electronics quickly and straightforwardly.

- Can you test only components or also complete assemblies?

Both. Although testing individual components takes up more time and the costs are higher, the results are very precise. With complete PCBs the costs and time to test are lower, but the inaccuracy is greater. The problem with complete assemblies

or entire PCBs is that in the case of a malfunction it may not be possible to localize the error precisely. Not all components are equally robust. Consequently you need to determine the ruggedness of the components first before you can develop a system for deployment in space. If you do this in the opposite way order, a single component may end up significantly degrading the viability of the entire assembly.

I have devised a new concept that lies between component and PCB level. In this way complete PCBs can be qualified more accurately.

- Who can I ask if I have any questions or a specific interest?

Jaime Estela

jaime.estela@gmail.com

jaime.estela@spectrum-aerospace.com

<https://www.linkedin.com/in/jaime-estela-9045b63b/>

Skype: jaimeestela

duced and thus relatively low-priced swarms of satellites, like GPS for example). Increased production-line manufacturing of satellites will create an even greater requirement for qualified electronic parts.

Space-COTS

For the NewSpace [7] movement of multiple (U.S.) startup companies aiming

to make space travel cheaper and more accessible, space-qualified COTS devices provide the means of getting commercial components qualified for use in space. The term and concept of 'space COTS' arose from work using commercial electronics in smallsat operations and also out of long-term involvement with ESA standards for qualifying electronic components. Space-COTS is a compromise

between non-qualified components (for example, in CubeSats) and fully qualified ITAR components. From the experience gained with small satellite missions it is known that many commercial components can definitely function in space for several years, despite not having been conceived for space applications. Space-COTS parts are tested at various levels and defined, by qualification level,

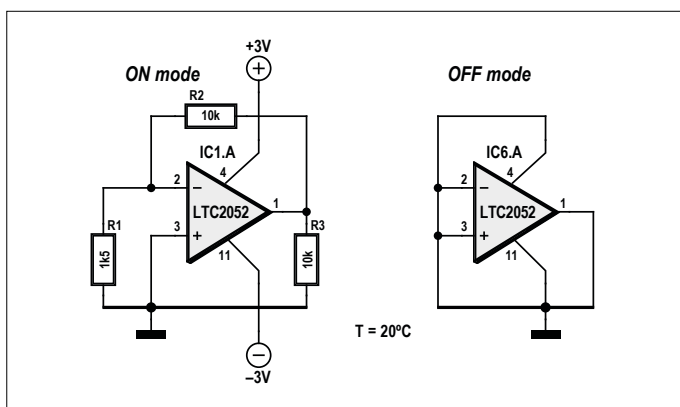


Figure 4. Two configurations of an LTC2052 for TID testing (source: Spectrum Aerospace).

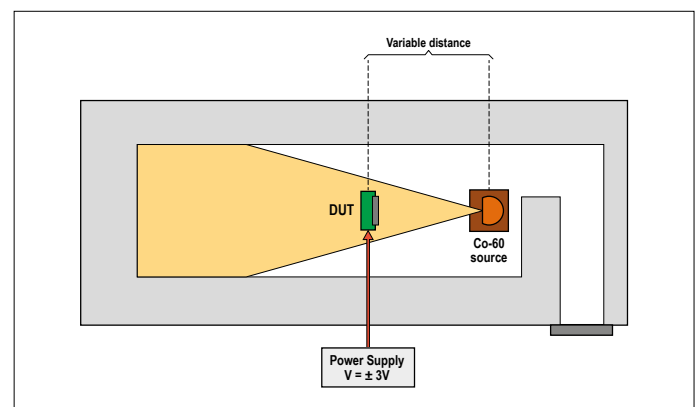


Figure 5. Test setup using a Cobalt-60 radiation source (source: Spectrum Aerospace).

in the following classes:

- Class A: Full screening test
- Class B: TID and SEE test (SEE = Single Event Effect)
- Class C: TID test

These classes accord with ESA and NASA specifications and also with the new ISO (International Organization for Standardization) standards.

StarLab

Many firms are developing new products for space flight. These products do not have any flight experience ('flight heritage'). To test a product in space, you need to invest at least €250,000 / £212,000 / \$274,000 (covering satellite plus flight but excluding the cost of test procedures). What's more, as well as the financial wherewithal, you must have the patience to resign yourself for a flight delay of at least 18 months. There are indeed dedicated satellites for technology proving trials but these launches take place very infrequently and are reserved for a very small number of privileged companies [8].

However, moves are now afoot to make the International Space Station (ISS) usable by a wider user base, enabling small private payloads to be installed on this platform. One of these initiatives is the StarLab project. One container can include multiple experiments; power and data connections are available for each experiment, with test data transmitted to Earth virtually in real time. The StarLab project, with its innovative qualification, provides regular access into space for everyman – and at moderate prices. Good electronics can definitely arise in the Maker community too. Their developments can also be of interest for space flight, even if the theme of space electronics is largely unknown in Maker circles. The StarLab project gives new electronics wings to pass tests in space. Testing your own designs in space is certainly still a big ask, but the StarLab project makes this elaborate and expensive exercise simple and advantageous. An entirely new field of activity opens up to developers and the space flight sector can profit from innovative, qualified products.

Is your curiosity aroused? Are you game

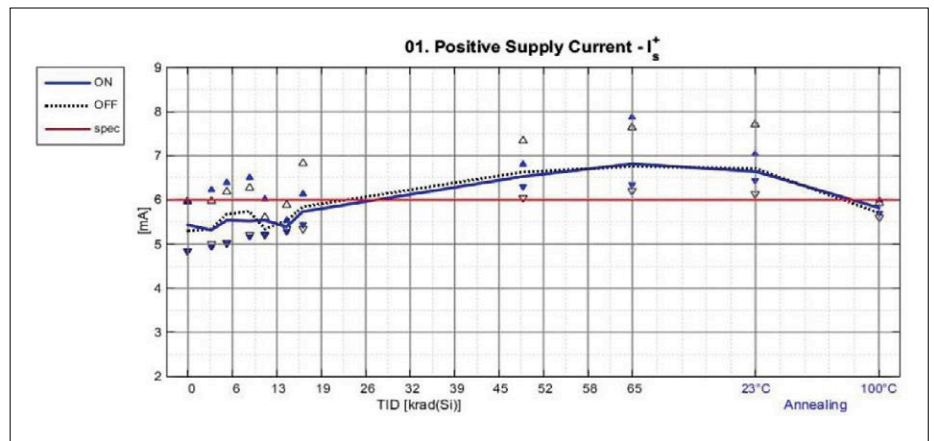


Figure 6. A test result: current consumption increases with increasing radiation dose (source: Spectrum Aerospace).

Space electronics: the bad old days

I guess the question I'm asked the most often is: "When you were sitting in that capsule listening to the countdown, how did you feel?" Well, the answer to that one is easy. I felt exactly how you would feel if you were getting ready to launch and knew you were sitting on top of two million parts — all built by the lowest bidder on a government contract.

— Astronaut John Glenn, the first American in orbit (1962), spoken in 1997.

for making experiments in space? Check out the answers to your burning questions about the StarLab project in the panel. The gateway to space stands wide open! ◀

(160036)

Sources and Weblinks

- [1] The history of space quality EEE parts in the United States, www.cti-us.com/pdf/HistoryEEESpacePartsinUSA.pdf, Leon Hamiter, Components Technology Institute
- [2] European Space Components Information Exchange System, <https://escies.org/webdocument/showArticle?id=167>
- [3] ESCIES Radiation List, <https://escies.org/labreport/radiationList>
- [4] ECSS-Q-ST-60-13C_Space Product Assurance, <https://escies.org/download/webDocumentFile?id=60887>
- [5] Nano And Microsatellite Markets Report, Northern Sky Research LLC, One Mifflin Place, Suite 400, Cambridge, MA 02138.
- [6] Small Satellite Market Observations 2015 – Forecast, Dr. John Bradford, Spaceworks Enterprises Inc., 1040 Crown Pointe Parkway, Suite 950, Atlanta, GA 30338 USA.
- [7] NewSpace - Wikipedia, <https://en.wikipedia.org/wiki/NewSpace>
- [8] DLR IOV/IOD Workshop, Marc Jochemich, DLR Space Administration, Brussels, September 2016.

“May I have the menu, please?”

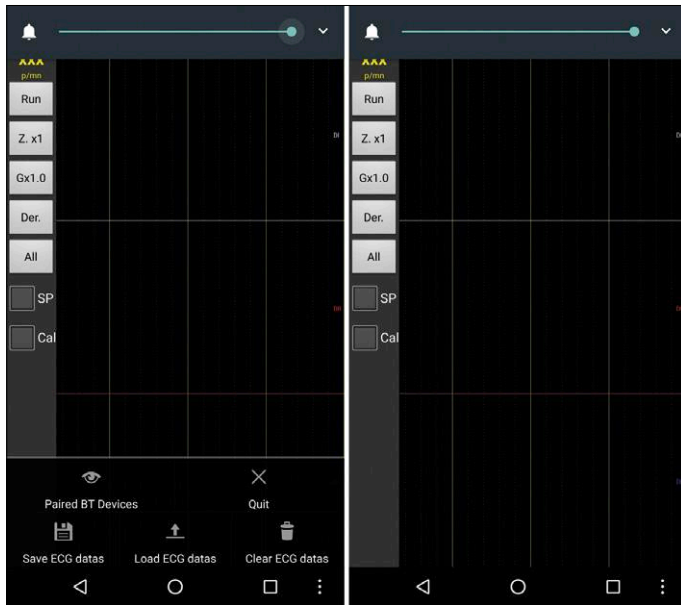


Figure 1. Screenshots of the cardioscope app, with and without buttons.

By **Luc Lemmens** (Elektor Labs)

It does happen occasionally that software stops working, particularly when that software has been around for quite a few years and doesn't want to function properly when running on the latest version of the operating system of a computer (system). The question we received at Elektor Labs about the app for the Elektor Android Cardioscope from 2013 initially looked to be one of these instances. But fortunately that turned out to be not the case.

One of our readers recently made a start on this project, but he couldn't find the buttons that should be at the bottom of the screen of the accompanying app, which are used to operate the cardioscope and are therefore essential for the operation of this circuit (**Figure 1**). On his particular mobile device, the Menu button that would have made these buttons appear, was missing.

Android devices (both tablet and smartphones) originally had three buttons (Home, Back and Menu) at the bottom of the screen for navigation, some even had four (the three aforementioned plus Search). On the really early models these were real pushbuttons, but these days it is more commonly a physical button for Home, flanked by two capacitive buttons. The various manufacturers have quite different views about the implementation, positioning and the icons used for these buttons. In **Figure 2** you can see a few examples of the Menu buttons on different smartphones, this overview is in all likelihood not even complete.

On some, more recent devices, the Menu button had to make way completely for the so-called Overview or App Switch but-

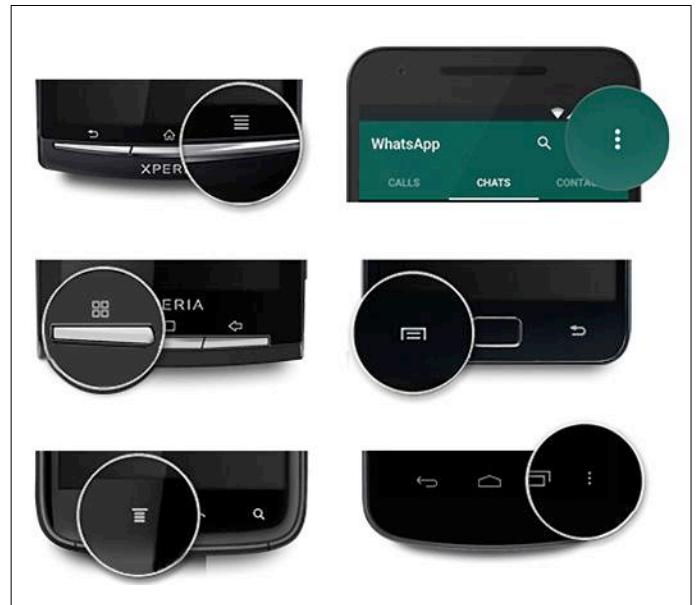


Figure 2. Different implementations of the Android Menu button.

ton, which the user can use to switch between the apps that are currently active. Our reader with the cardioscope project apparently had such a device and we did have to hunt around for a while before we found the solution. But fortunately this turned out to be very easy: the buttons on the screen for the cardioscope app appear when the Overview button is held for a slightly longer time. That is because the function of that button then changes from 'App Switch' to 'Menu'.

The question from our reader was answered reasonably quickly, but when you dig a little deeper into this relatively basic functionality, you will realize the enormous diversity — or is that proliferation — regarding the operation of Android devices. And then I remember again that last year I struggled quite a while before getting used to operating my new smartphone. Hereby my apologies to the people around me who may have heard the occasional expletive when I hit the wrong button again, and that is not even mentioning the people I have disturbed with unintended calls. ◀

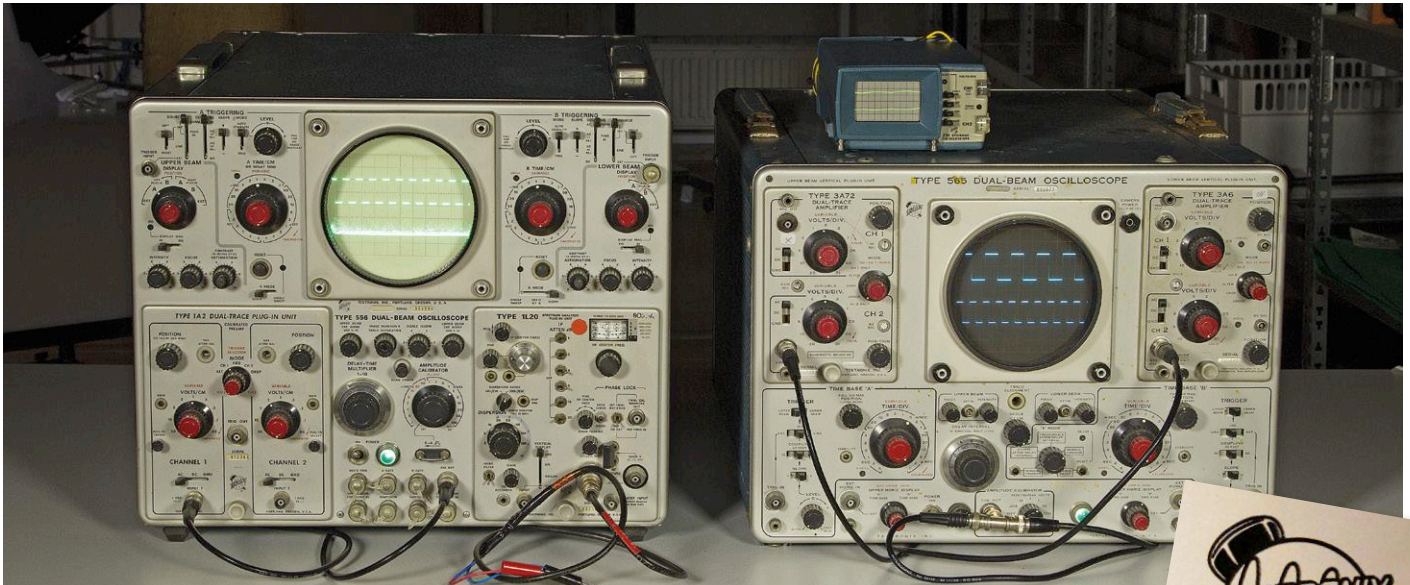
(160349)



Figure 3. Overview/App Switch buttons.

Tektronix 556 and 565 Dual-beam Oscilloscopes

The Triple-L beasts: large, loud, lumpy



By **Jan Buiting**, Retronics Resident Editor

“Oh my gosh look at that, and they boast *two* channels — hilarious! My old ‘scope used to do four or eight hands down!” It’s a common response from people first confronted with the Tektronix Types 556 and 565 cathode-ray tube (CRT) oscilloscopes, two of the largest, heaviest and most expensive instruments ever to serve electronicists in their labs if only for heating. These Goliaths are often regarded as the pinnacle of the tubed oscilloscopes from market leader Tektronix, although they contain more than a handful of semiconductors and there’s at least one Tek ‘scope that’s even heavier.

That’s dual-BEAM Sir

Many CRT oscilloscopes are loosely and indiscriminately described by their proud users as “1/2/4/x-channel”, “-trace”, “-way” or “-beam”. I estimate 98% of all CRT ‘scopes ever sold are single-beam instruments where the electron beam inside the CRT is chopped or alternated so rapidly your brain is tricked into perceiving two (or more) separate *traces*. This works fine for run of the mill signal viewing in the homelab using up to four traces, provided the CRT write speed is okay and the CRT itself is properly

designed — not a problem with most reputable ‘scope manufacturers as CRT technology in general can be said to have reached great heights.

Totally acceptable for daily work to this date, the rapid vertical alternating of the electron beam inside the CRT may however cause flicker and artefacts with really demanding signals, and even more so if the signals differ widely in repetition rate and/or frequency. Also, at a time when digital photography was in the realms of sci-fi, making a proper (Kodak Instamatic) screenshot of a mul-

tiple-trace ‘scope image was a challenge. That’s probably the reason for scientists and research institutions with deep pockets (like NASA) to prefer a dual-beam oscilloscope with a spec of 15 MHz over a dual-or quad-channel specified as “oh yeah much faster and much cheaper”. But if you require absolute and no-tricks observation of two signals on a single CRT face then it’s the dual-beam ‘scope you want. Sadly, these came at a premium due to the more expensive CRT with its double set of deflection plates, and in the instrument, the double V



Figure 1. Central area on the Tektronix 556 oscilloscope front with controls and connectors common to both beams.

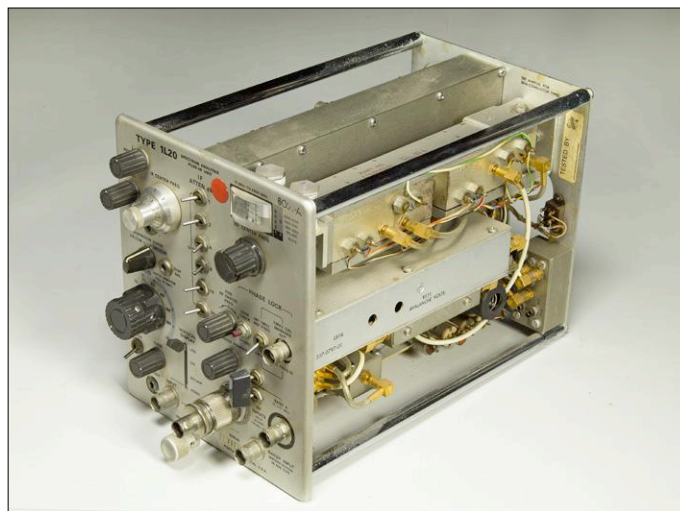


Figure 2. The Type 1L20 10 MHz - 4 GHz spectrum analyzer plug-in for 500-series Tek scopes is a rare bird.

amplifiers and more extra circuitry like two HV sections, and two timebases (though not often seen).

You really have to see it to believe it — the image on a dual-beam oscilloscope is extremely restful to the eye and no strange things happen when one signal is positioned over the other partly or fully. And yes, you can chop or alternate each of the *beams* to give 2 x 2 *traces* or even 2 x 4. Or observe a glitch in one period of a 50-Hz signal on the upper beam and ten or so periods of a 50-MHz signal on the lower beam.

Tektronix produced a limited number of oscilloscope models of the dual-beam variety: the 502, 551, 555, and 565 from the famous '500' series, the 7844 from the celebrated '7000' series, and the 5113 from the economy '5000' series. I know of at least one more manufacturer of dual-beam oscilloscopes and that's Philips with their transistorized PM3230, PM3231, PM3232, and PM3233, and the tubed PM3236. Let me know if you are aware of other manufacturers. I have never seen a triple- or quad-beam oscilloscope, have you?

Special delivery! One Tek 556

Although Elektor's staff photographer Mart has known *Retronics* for many years for its out of the ordinary, laborious but amusing photo assignments, the 45+ kilograms (100 lbs) "objects" politely suggested by me for this installment would have ravaged his expensive photography table designed to carry Arduinos, RPi's and other minute electronics. By coincidence I had a stronger 160 x

80 cm office table available retrofitted with swivel & lock castors and this was able to carry the 556 and 565 monsters as well as some tools, cables and a 214. So I wheeled in the objects.

The **head image** shows the two Goliaths and one David (a 214) in action, jointly drawing about 1,600 watts from a single 230 VAC AC outlet in our former photo studio. The big fans in the 556 and 565 made a racket. The 556 (left) is displaying its 1-kHz calibrator signal on the upper beam, and on the lower beam the RF spectrum of the 2.4-GHz Wi-Fi band at Elektor House, picked up basically on two banana plugs.

Although the big 556 is a dual-beam all-symmetrical design with many duplicate circuits, it does have controls and connectors common to both beams. These are conveniently arranged in a central area pictured in **Figure 1**. This

wonderfully preserved 556 is S/N 004584 from Tek's Portland, Oregon factory.

The 556 is a configurable 'scope with two bays that accept a wide range of plug-in units to choose from. Here I am fairly randomly using a Type 1A2 50-MHz dual-trace vertical amplifier (left) and a Type 1L20 10-4200 MHz spectrum analyzer pictured separately in **Figure 2**. The lower-trace signal proved difficult to photograph due to the camera distance and the non-periodic image due to our busy computer and smartphone users! In an attempt to indicate the sheer size of these instruments using a printed medium like this A4-size (approx. US letter) magazine page I opened the 556's top cover and placed a 9-volt battery on its CRT cover (**Figure 3**). I could have put 20 there. A closer look at the CRT connections is afforded by **Figure 4** — specifically the five connections for the



Figure 3. Inside top view of the Tek 556 and its massive dual-beam (not: dual-trace) CRT. It's a good thing we now have LCD and LED screens on our 'scopes.

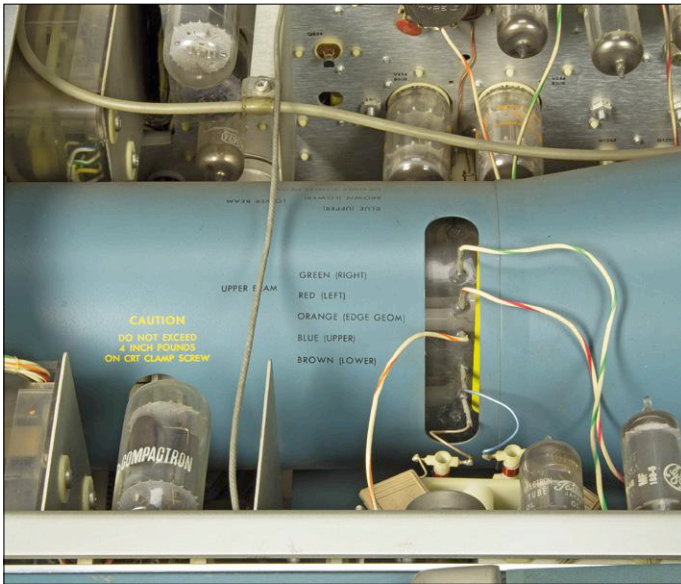


Figure 4. CRT connections in the 556. Utterly simple wires, push-on receptacles, and through-glass pins are used.



Figure 5. Tektronix 3A6 dual-trace plug-in. Two of these in a 556 'scope give you four crisp traces from two electron beams.

upper-beam plates. An identical set of five wires for the lower beam is at the opposite side of the CRT neck (the print is just visible). Note the cool use of the words "upper" and "lower" to denote the vertical deflection plates. Also, the two end compensation coils for the V plates, and the famous Compactron tube! This 556 required no repairs, just minor cleaning. It came with the two plug-ins pictured, to which I was fortunate to add

a 1L10 and a 1L30 later, which gives me coverage from 1 MHz to 10.4 GHz.

The 565 with trouble

Compared to the 556 the slightly smaller and lighter 565 is marked by even more options for triggering, as well as time-base separation, -sharing and -crossing. The 565 offers 'just' 15 MHz bandwidth compared to 50 MHz for the 556, which makes one wonder why Tek went

through the trouble of producing it. The answer is probably to extend the popular 560-series of scopes with a dual-beam instrument. The 560 sub-series employs special plug-in units designed to drive a CRT's V and H plates directly.

Figure 5 shows the 3A6 dual-trace amplifier plug-in unit used in my 565. Comparing it to a 500-series plug-in like the 1L20 (Figure 2 and the head image refer) you can see it's narrower. At the rear side sit two fat E55L 'SQ' (special quality) tubes which handle the driving of the V plates in the 565 main instrument at up to 10 MHz and low capacitance to avoid losses and distortion. The range of 560-compatible plug-ins is also impressive, with 4-trace, 8-trace, GHz sampling units — a feat at the time and again challenging the designers at Hewlett Packard!

Although it came from a different owner, the 565 was transported to its new home in my Suzuki SX4, along with the 556 and a boxful of plug-ins. Before it was even allowed into my garage I had to give it a quick clean as the unit was positively filthy after about 20 years in an industrial cellar along with 30 tons of e-debris. I started up the 565 on a variac and after a good four hours, on reaching about 185 VAC (of 230 nominally), I was happy to see a trace appear. Just as I wanted to position the mesmerizing blue trace toward the center of the screen, a mini explosion was heard, and the trace disappeared. I switched off immediately but a vile smell soon filled the room and gradu-

Tektronix 500-series dual-beam oscilloscopes			
Type	Weight (less plug-ins)	Main features	Elektor Retronics Collection item
502(A)	23.6 kg (52 lb.)	500 kHz; high sensitivity; low cost; compact; 1x sweep; fanless.	✓
551	Indicator: 23.6 kg (52 lb.) PSU: 19.2 kg (42 lb.)	30 MHz or lower at high sensitivity; 2 x V plug-in; 1x sweep; no delay, separate power supply; 1x fan.	✓
555 "Triple Nickel"	Indicator: 31 kg (68 lb.) PSU: 20.5 kg (45 lb.)	30 MHz or lower at high sensitivity; 2 x V plug-in; 2 x H plug-in; delay; separate power supply; 2x fan.	✗
556	40 kg (88 lb.)	50 MHz; 2 x V plug-in ("letter" and 1xx series); 2x sweep; integrated power supply; 1x fan.	✓
565	31 kg (68 lb.)	15 MHz; 2 x 560-series (Type 2xx/3xx) plug-in; 2x sweep, delay; 1x fan.	✓

ally my entire house. I feared the demise of an electrolytic capacitor somewhere in the instrument but a visual inspection did not provide any clues. Also, no fuses were blown, and the 'scope could be switched on again with all tubes glowing and the five regulated voltages measuring within 2% i.e. beyond reproach. After sniffing around inside the scope I removed the cover on the HV section and spotted the evildoer: a 'black beauty' capacitor with signs of an eruption. Impressive, the smell from such a small device and an even smaller hole! The capacitor is part of an oscillator circuit at the heart of the regulated high-voltage converter in the 565. Ten minutes later a 0.1- μ F 630-V replacement capacitor was in place (**Figure 6**) and after resetting the controls on the 565 to their default settings, two very crisp traces appeared on the CRT screen. That black Sprague 1 nF / 1000 V capacitors just visible under my pencil is a likely candidate for replacement too but it's okay for the while.

I noticed both traces were too short at 7.5 cms instead of slightly over 10 cms. This was remedied by adjusting the SWEEP WIDTH and associated x1 GAIN pots inside the instrument, but requires further investigation and recalibration with a time mark generator. The presets had a "dusty" feel to their adjustment and were duly IPA'd.

My instrument has a blue-phosphor CRT and an extremely fine-mesh screen cover that makes the CRT face look like a black hole but effectively brings out the traces beautifully and without signs of glare or over radiance of the beams even at high intensity settings (separately for the two beams ☺). Also note the 6.3-V camera power connector! Mart loved the matt black mesh at it allowed him to photograph with ease the internal calibrator waveform with **individual** timebase settings for each of the **two** traces beams for the purpose of reproducing in **Figure 7**.

Conclusion

These instruments have so many options they easily take 15 minutes to get the settings right and allow the measurements to commence. Consider having two beams (Upper and Lower) writing for example, two traces each (Upper-upper and Upper-lower; Lower-upper and Lower-lower), two timebases A and B, delayed and non-delayed, applied to Upper and/or Lower, with triggering

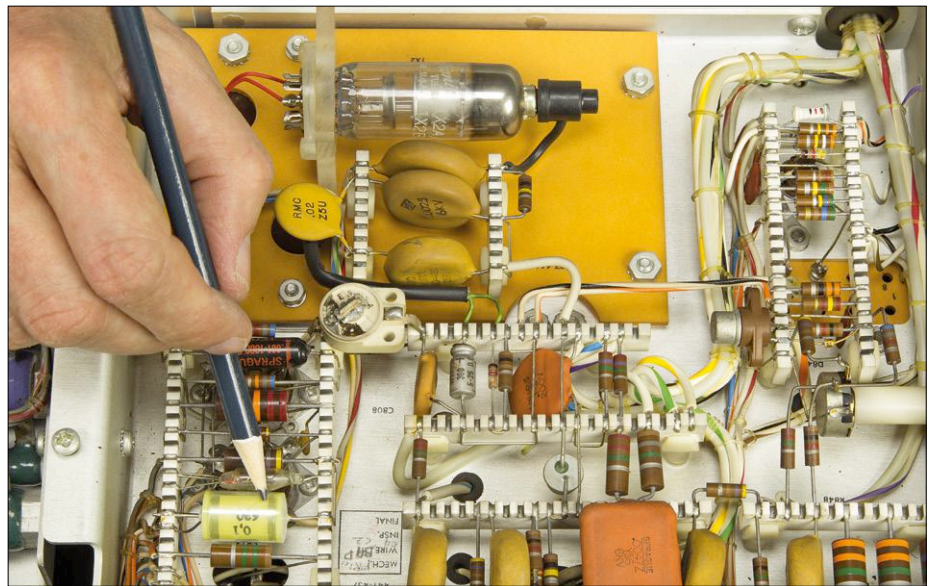


Figure 6. The CRT high voltage section in my 565 came alive again with a new capacitor fitted.

occurring on A or B, or the signals cross-routed... And how can we forget the fan noise and about 500 to 700 watts of heat to endure? Or their backbreaking weight? But then the look and feel of these instruments is beyond anything offered by present-day 'plastishine' oscilloscopes. Neither the 565, the 556 or the associated plug-ins discussed here were calibrated as they should have been after so many years of neglect. They also deserve further cleaning and optimizing but I wanted to show you the instruments in action in the first place. Sadly, switching them on again will not be possible for some time to come as the beasts will go into storage at Elektor Warehouse while Elektor completes their move to new offices in Aachen, Germany. Be careful, Mister forklift driver! ◀

(160402)

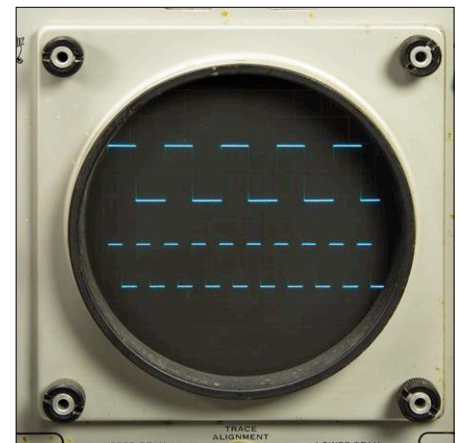


Figure 7. Tektronix oscilloscopes are known for their extremely sharp traces and outstanding pulse response. This image photographed on a 565 with a special fine-mesh screen filter drives the point home. Note: instrument not yet calibrated.

EST[®] 2004

www.elektor.tv

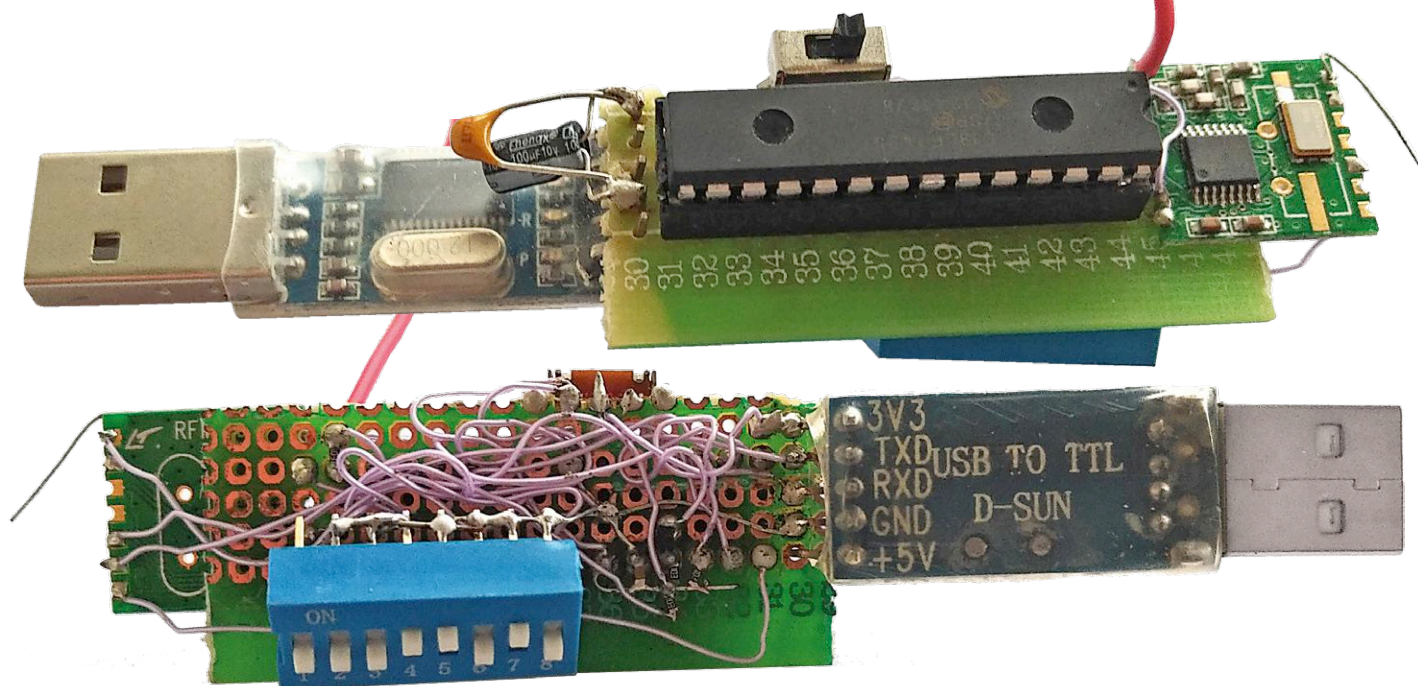


Retronics is a regular section covering vintage electronics including legendary Elektor designs.

Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com

Sniffer using RFM12 Radio Module

spy on the ether



By **Laurent Labbe** (China)

Several years ago, I designed a thermometer with Nixie display using remote sensors. The thermometer displays the indoor and outdoor temperatures as well as the time, captured from an external GPS module. A specific application checks a remote mailbox on a regular basis (counting the passes of the mailman). All the sensors send their data (wirelessly) for display every 30 minutes. For various reasons, I could not connect the PC to the display box to capture the arrival of the data frames and debug them. For this reason I made use of an RFM12 radio module to spy on my thermometer. Now that the software is up and running, I'll share my experiences with you.

To be sure to see all the data sent to the thermometer display, I developed a simple tool: the RFM12 sniffer. It captures all that passes through the air and sends it via a USB connection to a PC which displays the data. The first version of my device was intended to capture the data frames from the thermometer formatted to a (simple) proprietary protocol, with a header and a checksum. Here, the principle was very simple: display on the PC everything that passed over the 'ether' without any processing on the data structure, or verifying the control sum.

Two essential ingredients: a PIC and a radio module

To run my device, I chose a PIC 18LF2420 microcontroller which runs on 3.3 V and an internal clock of 8 MHz, but you could use any other microcontroller. For power, the USB serial module used has two outputs: 5 V and 3.3 V. As the RFM12 needs 3.3 V, we will use that option with the correct filtering. For information, I often use USB-to-serial modules of the Prolific brand.

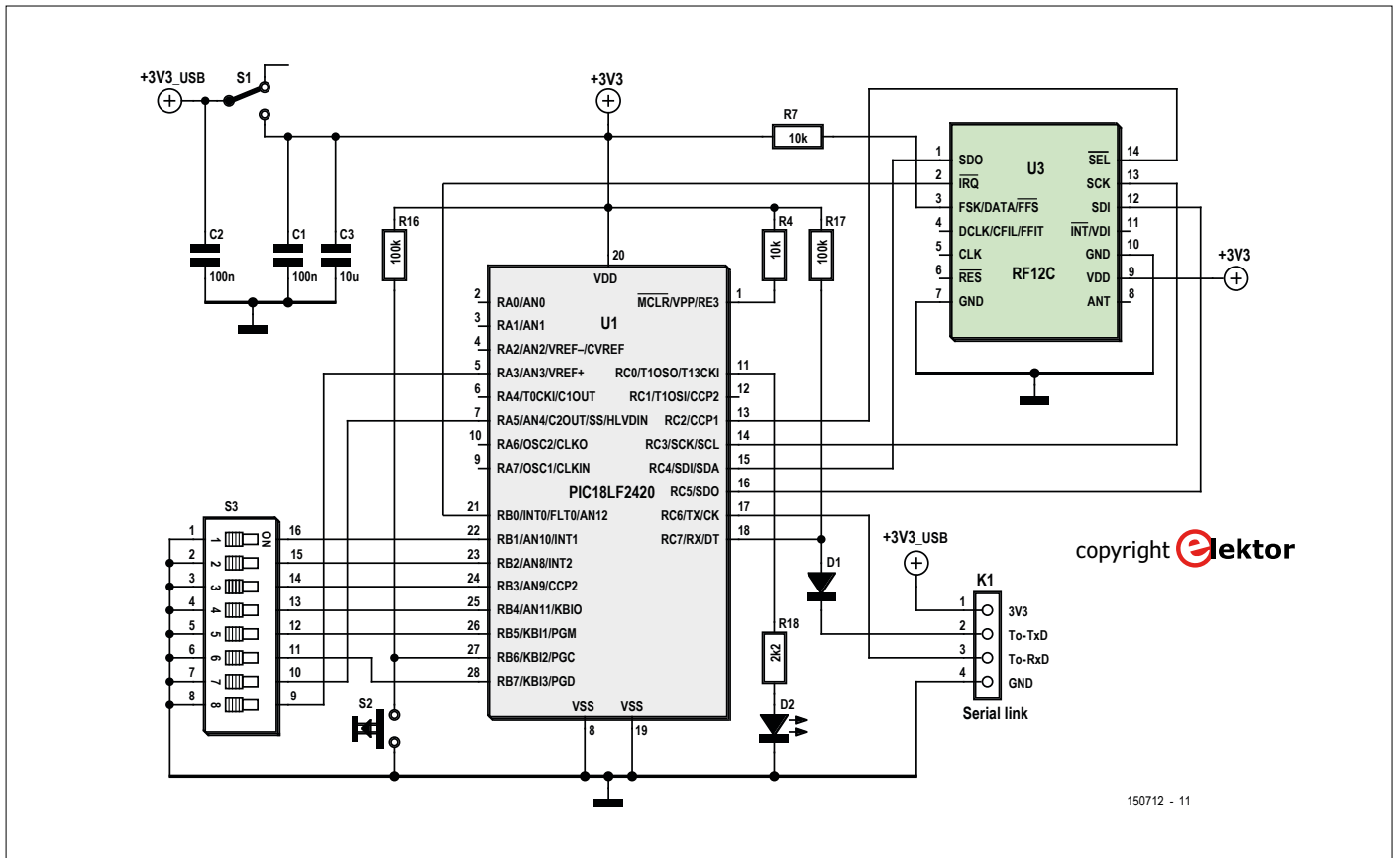


Figure 1. Schematic of the RFM12 sniffer.

To adapt the levels between the μ C (3.3 V) and the USB (5 V), the TX pin (17) of the μ C is connected direct to the USB connector while the Rx pin (18) is protected by a diode and equipped with a pull-up resistor.

A bank of DIP switches (S3) allows several parameters of the circuit to be set. The first two switches define the rate of transmission to the PC (see **Table 1**).

If the RFM12 module receives nothing further, switches 3 and 4 set the delay after which the PC receives an automatic line feed (CR/LF) (see **Table 2**).

Note: switches 7 and 8 should not be used. These switches are connected to Port A without any pull-up resistors, and Port A does not have any internal resistors.

Switch S1 (run/stop) is primarily to reload the software without having to pull the μ C off the board. The pushbutton S2 is to restart the program.

The resistors are mainly pull-up resistors: R4 for reset of the

μ C, R16 for the pushbutton S2 and R17 for the serial link on connector K1. The bank of DIP switches uses the internal pull-up resistors of Port B. LED D2 flashes when data is received.

A morsel of code

The code is written with MikroC (download at [2]). The routines for the RFM12 are the most sensitive. I adapted routines found on the internet. I added, for example, a watchdog timer while waiting for data. This code is a stripped-down version of that used in my Nixie Thermometer. Some parts of this code are thus not used for the sniffer presented here.

The main loop is used for waiting for valid data: the IRQ output of the RFM12 alerts the μ C to the arrival of data (input PORTB.F0), following which the data is transmitted to the μ C (PORTC.F4) to be sent to the PC via the USB link (PORTC.F6). I noticed that the RFM12 often sent erroneous data. It is thus necessary to reset regularly the receiver and the FIFO register.

Table 1. Transmission speed to the PC (S3)

Pos. 1	Pos. 2	Pos. 1	Pos. 2	Pos. 1	Pos. 2	Pos. 1	Pos. 2
OFF	OFF	ON	OFF	OFF	ON	ON	ON
1200 baud		9600 baud		19200 baud		19200 baud	

Table 2. Silence time before a carriage return / line feed (CR/LF) is sent (S3)

Pos. 3	Pos. 4	Pos. 3	Pos. 4	Pos. 3	Pos. 4	Pos. 3	Pos. 4
OFF	OFF	ON	OFF	OFF	ON	ON	ON
no CR/LF		1 s		10 s		20 s	

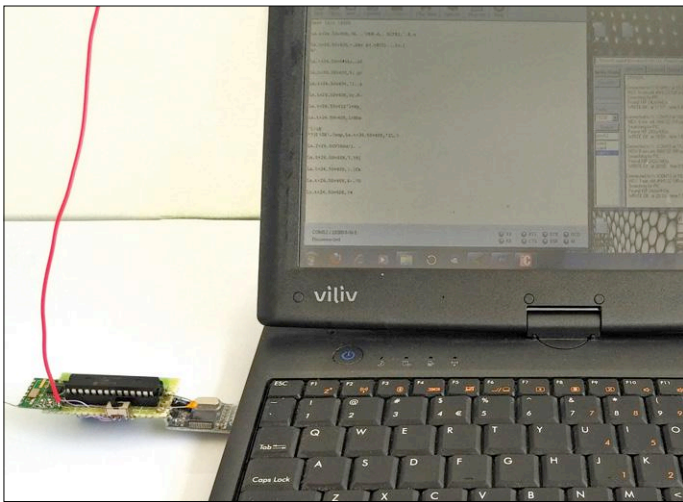


Figure 2. The spy in action: the device connected to a laptop PC.

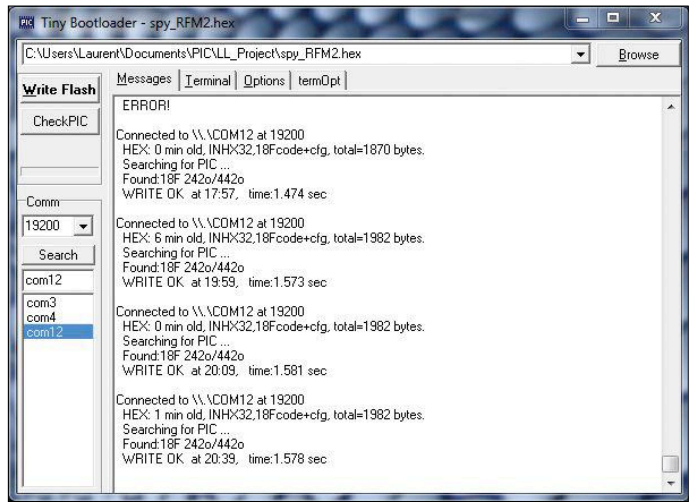


Figure 3. Bootloader screen capture.

Here, I have used DIP switches 5 and 6 to program a delay (see **Table 3**); the receiver and the register are automatically reset when the timer times out. Without this, the RFM12 continues to receive data after each frame.

Usually, I use a bootloader for all my applications with the 18LF2420. For me, the best one is the **Tiny Pic Bootloader** (see [3]). It is not compatible with all the PIC μ Cs, but it works with the 18LF2420. Since the 18LF2420 cannot dynamically change the watchdog timer frequency, you can perhaps recompile the loader to set this frequency. The code used is around 100 bytes and is reliable.

I'm listening

There's no magic needed to use the assembly: just connect it to a USB port of the PC. You must first have installed the driver from the manufacturer of the Serial USB module that you've chosen. Then launch a tool like HyperTerminal or CoolTerm and set the correct speed. Some data should be shown. If the timer (Table 2) is active, the new lines should improve readability. With the save function you are able to leave the sniffer running for several days and analyze the captured frames later. With CoolTerm, the display of characters is frozen from time to time, but if you examine the hexadecimal data, everything is there. ◀

(150712)

Web Links

- [1] Elektor Labs project:
<https://www.elektormagazine.com/labs/sniffer-for-rfm12-display-exchanges-over-the-air-on-pc>
- [2] Source code, compiled code, suggested PCB artwork:
www.elektormagazine.com/150712
- [3] Tiny Pic Bootloader:
www.etc.ugal.ro/cchiculita/software/picbootloader.htm

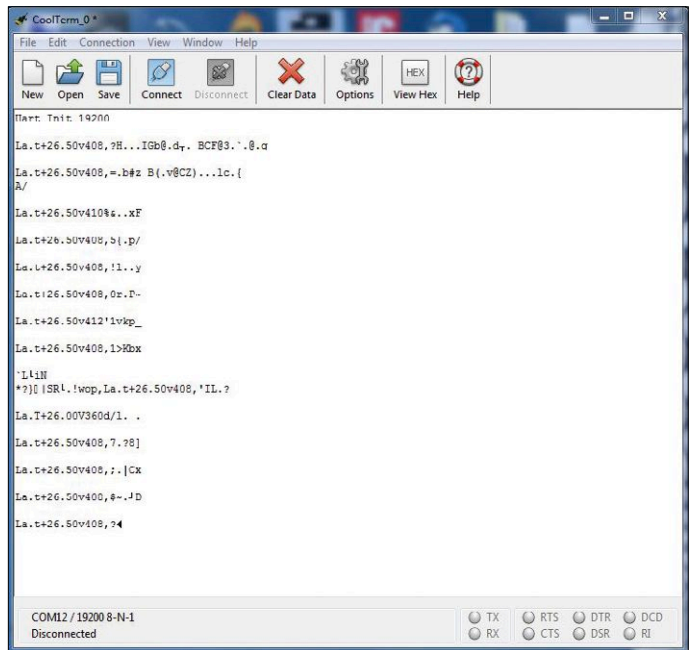


Figure 4. Terminal screen capture.

Table 3. Expected frame length (S3)							
Pos. 5	Pos. 6	Pos. 5	Pos. 6	Pos. 5	Pos. 6	Pos. 5	Pos. 6
OFF	OFF	ON	OFF	OFF	ON	ON	ON
10 bytes		20 bytes		40 bytes		100 bytes	



Tips and Tricks

From readers for readers

Here is another neat solution to a tricky problem

Test Pattern Generator using a Digital Camera

By Alfred Rosenkränzer (alfred_rosenkraenzer@gmx.de)

For the purposes of setting up CCTV surveillance systems I needed a small, mobile, battery-powered video test-card picture generator. In my work as a video technology developer, I use several different types of test signal generators but they are neither small nor battery powered. I started to think about how I could build a portable generator out of a combination of components such as a CPLD, RAM and DAC.

It wasn't long before I realized that I (and more than likely, you also) already own a portable device with all the necessary parts: A digital camera! Not only does it have a display to show images snapped by the camera but also a jack to output a video signal for hookup to an external monitor. Older cameras 'only' have an analog output (often via 3.5 mm jack); newer models use an HDMI interface. There are ready-made test card images on the Internet, for example at Burosch [1]. Unfortunately, none of my cameras could read image files that it (or a similar camera model) had not taken itself. Attempts to identify and modify the necessary Exif metadata associated with the image proved fruitless. If any reader has been more successful than me making the file modifications, I would be grateful for their help.

To work around the problem, I simply took a photo with the camera and then pasted the test card image using an image processing program. For the analog output signal a low resolution 640 x 480 or 800 x 600 pixel image is good enough. The video standard for PAL systems is 720 x 576 and for NTSC 720 x 480 pixels. **Figure 1** shows the resulting test pattern on my three cameras.

In order to measure the frequency response of a video cable run (in my case using a cable length up to 240 ft.) you can use a special video test image called multiburst. As the name implies, it consists of several bursts, starting with low frequency, up to the upper limit frequency of the system (about 5 MHz for PAL and 4.2 MHz for NTSC). A multiburst pattern used to be integrated in the standard test card, which was shown on TV during breaks in broadcasting. At the end of the transmission path you can observe the test pattern using a display monitor or check the signal with an oscilloscope. If all frequency bursts have the same amplitude it indicates that the cable is not attenuating the signal and has sufficient bandwidth. The bursts can be displayed in



Figure 1. The test card on my three cameras.

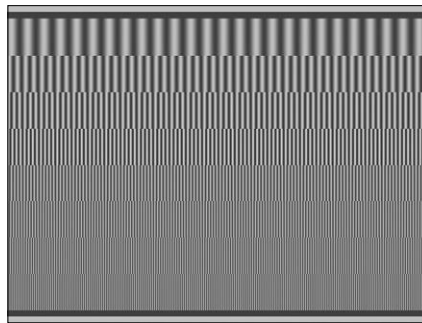


Figure 2. My custom Multiburst test card.

one line or distributed over the image from top to bottom. This is referred to as horizontal or vertical multiburst. I wasn't able to find a suitable test card image so I created one myself (interested readers can contact me by e-mail for more details). Along the top and bottom are reference strips to represent the maximum (light) and minimum (dark) levels. The start and end of each line blends to grey to reduce the chances of signal over-shoots causing problems. The resulting test card is shown in **Figure 2**.

The internal D/A converter and post filtering in a standard camera limits picture quality so it will never be as good as a professional test image generator. The maximum signal frequency and picture quality suffer because of the reduced sampling rate and will not always be high enough to occupy the full 5-MHz spectrum. The image files I used can be downloaded from the Elektor Magazine web page [2] associated with this article. If your cameras are as fussy and uncooperative as mine, you can, as described above start with an image produced by your camera and paste the test card image onto it, saving the result as a new file using Photoshop or similar tools.

Web Links

- [1] www.burosch.de/testbilder-uebersicht/311-first-check-full-hd-testbild.html
- [2] www.elektormagazine.com/160426



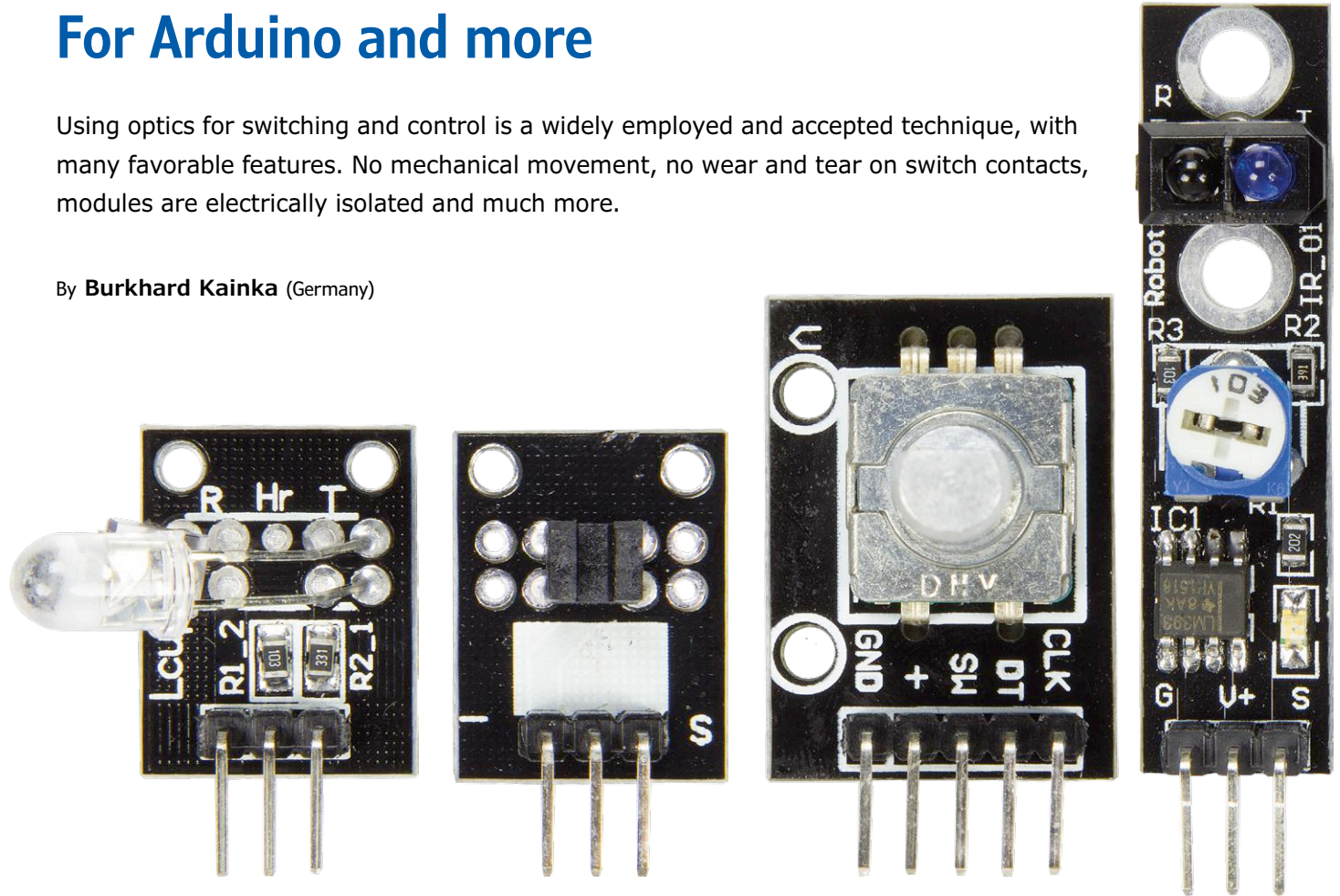
Have you come up with an inspired way of solving a really challenging problem? Or found an ingenious but 'alternative' way of using some component or tool? Maybe you've invented a better or simpler way of tackling a task? Do write in – for every tip that we publish, we'll reward you with UKP40 (or local equivalent)!

Sensors Make Sense (4)

For Arduino and more

Using optics for switching and control is a widely employed and accepted technique, with many favorable features. No mechanical movement, no wear and tear on switch contacts, modules are electrically isolated and much more.

By **Burkhard Kainka** (Germany)



This time our subject is optical switches and their applications, from fork sensors to reflex light barriers and pulse detectors. We'll cover the best methods of interpreting and analyzing the signals too. Once again we are using sensors from the 35 Sensors Kit, available direct from Elektor [1].

As always, all sample programs and listings can be downloaded from the web page for this article [5].

The tracking sensor

We're all familiar with little robots on two wheels that can

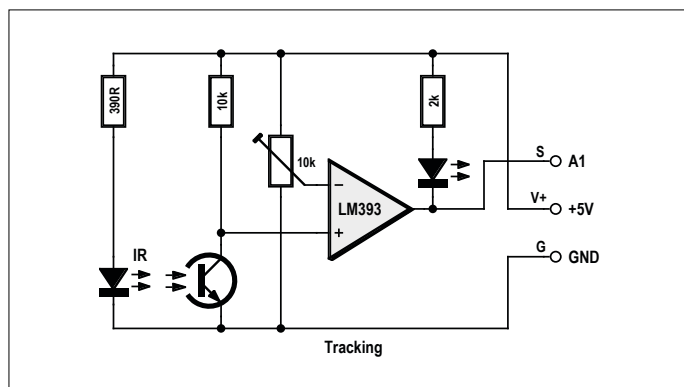


Figure 1. Adjustable reflected light barrier.

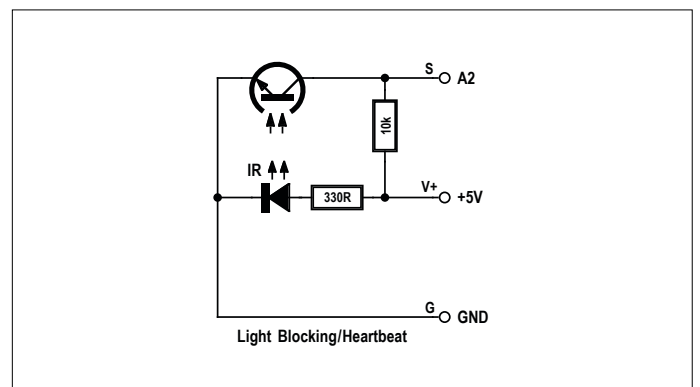


Figure 2. Fork sensor.

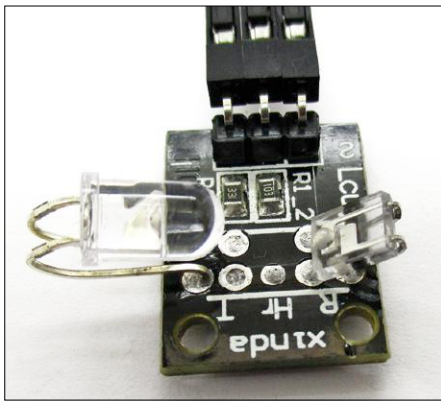


Figure 3. Assembled pulse sensor.

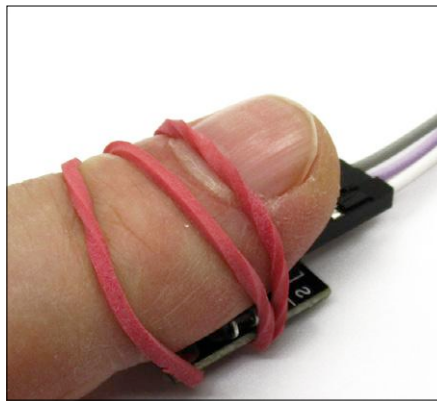


Figure 4. The pulse sensor in action.

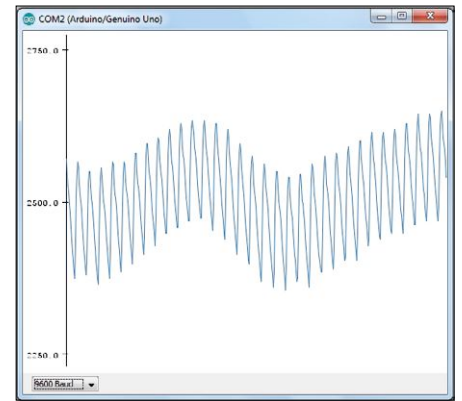


Figure 5. Voltage on the pulse sensor.

follow a white line. These assess work by assessing the reflectivity of the surface below them. An IR transmit diode illuminates the surface beneath them and an IR phototransistor detects the reflected light. In basic principle terms this uses what is called a reflex (or reflected) light barrier. To adjust for optimum sensitivity for differing surfaces and clearances the tracking sensor uses a potentiometer (pot) and a comparator (**Figure 1**).

Thanks to the LED on the output of the comparator we can easily test its functionality. Your finger will provide the reflective surface if you hold it a centimeter away from the sensor. You can now adjust the pot to get the best switchover point. A digital signal will appear on the output and this can be analyzed and processed in the usual way. You can either use the software described in part 2 of this series [3] or you can connect an actuator direct. What you do with this after that is left to your own imagination! It doesn't have to be a robot even. Perhaps you need a light or a fan that is switched on by a hand gesture, or you may have some other requirement.

Optical fork sensors

In many machines and devices (from scanners to 3-D printers) you will find limit switches for determining the position of a moveable component. Mechanical contacts or microswitches often suffer problems concerning their long-term stability and need to be replaced after fairly long periods of use. A fork sensor gives better service because the IR diode and phototransistor used display no signs of ageing, at least with moderate LED current.

A transmit diode and phototransistor (**Figure 2**) are positioned close together. The fork-shaped housing contains a slot, in or through which you can place or pass an object that is opaque to light. The on/off switching transition achieved is easily reproducible with an accuracy of less than a millimeter. Fork sensors of this kind are used also in computer mouse devices and demonstrate their reliability day in and day out.

The output signal at the collector of the phototransistor can be polled either analogly or digitally, according to your choice. Exactly what is done with the signal then depends on the particular assignment and the software. Your starting point can be with an accuracy of less than 1 mm — possibly exactly this, as required for a 3-D printer. In less critical cases a purely digital (yes or no) query will suffice: light or no light.

The pulse sensor

Heartbeat monitors and other pulse meters frequently use an optical procedure. Light of an appropriate wavelength is passed through a finger or an earlobe and then received using a phototransistor. Part of the light is absorbed in the blood inside the body. The heartbeat varies the momentary blood flow periodically and consequently modulates the light stream to a small degree. Naturally these signals must be evaluated analogly, as we are not expecting any major variations.

The heartbeat sensor is constructed around a fork sensor (see Figure 2), but with a greater distance between transmitter and receiver. Both the IR transmit diode and the phototransistor have long connecting leads, enabling you to adjust the optimum position. The PCB is identical to that used for the fork sensor, except that the connector and components are fixed on the other side.

In use you will need to bend the IR diode and the phototransistor so that a finger can be clamped between them both. It's also vital that the finger cannot touch any conductive area on the PCB, because additional mains (AC line) hum interference could be picked up in this way. On our example the sensors were flexed as seen in Figure 3. The IR LED then shines almost directly onto the phototransistor. The output voltage is now close to zero. If you next lay your finger in the gap, the light is shaded. The phototransistor conducts less, meaning that the voltage rises. For longer tests it makes sense to hold the sensor onto the finger with a rubber band (Figure 4). It is, you see, important that the finger is relaxed and the signal is not falsified by varying pressure.

For your first test with the Arduino IDE the program VoltageAD2 from part 1 of the series [2] will be fine. The voltage on the phototransistor can now be represented using the serial plotter (Figure 5). The light modulation caused by the pulse is clearly visible. Overlaid on the signal are slower variations, caused by unavoidable movement. This explains why evaluating the signals can be a challenge.

To improve the interpretation we first used a lowpass filter (Listing 1). This simply produces a moving average of the value measured. For this we need nothing more than a couple of lines of code. Each of the current values measured is added to a sum total of average values mean that is afterwards reduced by 1/20. Each individual measurement can then influence the average by only 5%. The time constant of the lowpass filter

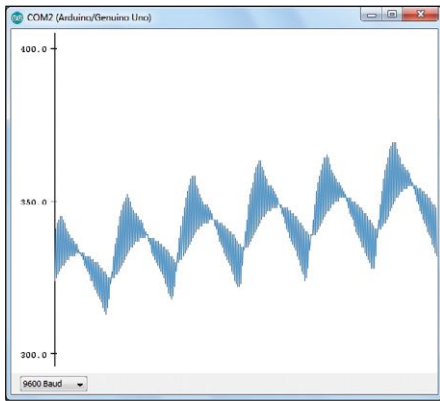


Figure 6. Original and average values.

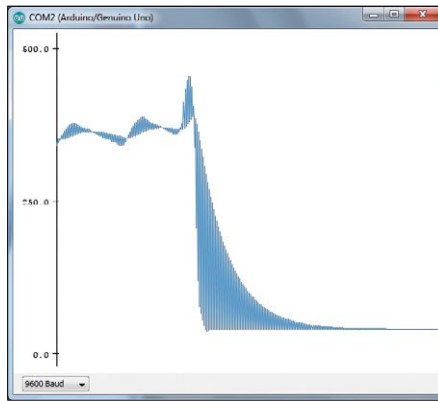


Figure 7. Pulse response.

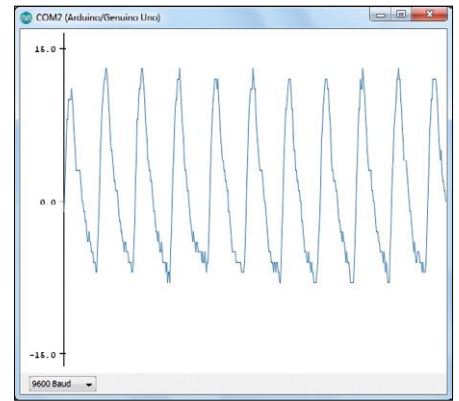


Figure 8. Filtered pulse signals.

Listing 1. Generating an average.

```
//VoltageAD2 0...1023 at AD2 filter
int sensorPin = 2;
int value;
int mean;

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
}
void loop() {
  value = analogRead(sensorPin);
  mean = mean - mean / 20;
  mean = mean + value;
  Serial.println(value);
  Serial.println(mean / 20);
  delay(19);
}
```

Listing 2. Highpass filter.

```
//Filter2AD2 0...1023 at AD2 filter
int sensorPin = 2;
int value;
int mean;

void setup() {
  Serial.begin(9600);
}
void loop() {
  value = analogRead(sensorPin);
  mean = mean - mean / 20;
  mean = mean + value;
  value = value - mean / 20;
  Serial.println(value);
  delay(19);
}
```

Listing 3. The Arduino pulse meter.

```
void loop() {
  value = analogRead(sensorPin);
  mean = mean - mean / 20;
  mean = mean + value;
  value = value - mean / 20;
  if ((old < 0) & (value > 0)) {
    time2 = millis();
    pulseTime = time2-time1;
    time1 = time2;
    //Serial.println(pulseTime);
    n = n + 1;
    pulseFreq = 60000 / pulseTime;
    if (pulseFreq > 45) {
      Serial.println(pulseFreq);
      lcd.setCursor(0, 0);
      lcd.print(pulseTime);
      lcd.print(" ms ");
      lcd.setCursor(0, 1);
      lcd.print(pulseFreq);
      lcd.print(" /min ");
    }
  }
  old = value;
  delay(20);
}
```


is therefore the sensing period times 20, in this case 0.4 s. The sampling rate is fixed at 50 Hz in order to attenuate any interference signals from the AC line current (of course you would use 60 Hz where this is the norm). The cutoff frequency of the filter is $f = 1 / (2 \text{ Pi } T)$, in other words around 0.398 Hz. For illustration purposes the unfiltered and filtered measurements are always shown alternately.

On the serial plotter you can now see how the filter works. The moving average follows the signal with a degree of inertia (Figure 6). If you remove your finger rapidly from the sensor, you can observe the pulse response of the lowpass filters (Figure 7).

All that remains is to generate the difference (input signal minus the average) (Listing 2). You do this by creating a high-pass filter and then obtain a clean pulse signal that is rid of all the sluggish variations. With this done you can now set about evaluating the pulse frequency.

As well as this you can now recognize some further details. The signal always rises steeply and drops back more gradually (Figure 8). You can see how, with every beat, the heart pumps blood at high pressure through the arteries into the finger, from which the return flow through the veins takes longer. You can also spot easily any irregularities that may occur in the pulse rhythm.

To make a complete pulse meter device out of this now we need to measure the intervals between the positive flanks (leading edges). The Arduino has a simple timekeeping function by the name of `millis()`. This returns the time elapsed since the last Reset operation in milliseconds. In this way you can measure the interval between two heartbeats. The program (Listing 3) indicates the pulse time in ms and the pulse rate in beats per minute on the LCD of the Elektor Extension Shield [6]. The frequency is calculated from the pulse duration for each individual heartbeat, meaning that you don't have to spend

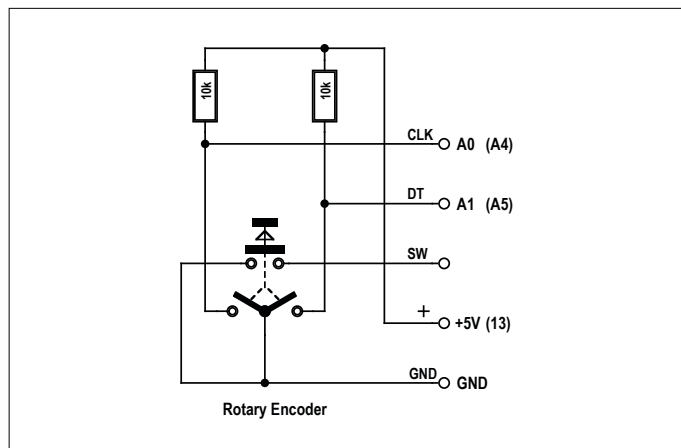


Figure 9. Rotary encoder.

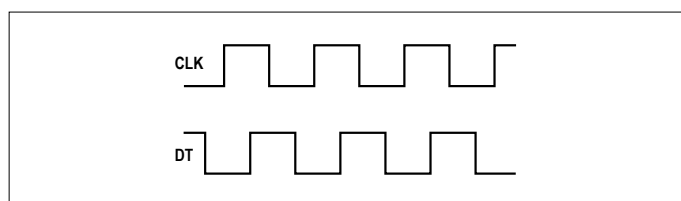


Figure 10. Switching signals during rotation.

long waiting. It does nevertheless happen that a pulse can go missing, even after slight movement, which can then lead to double or threefold pulse timings. This is why measurements are output only when the pulse frequency is greater than 45 beats per minute.

Listing 4. Bascom pulse meter.

```

'-----
'Pulse.BAS  ADC2, LCD
'-----

...

Config Timer0 = Timer , Prescale = 1024
On Ovf0 Tim0_isr
Enable Timer0
Enable Interrupts

...

Do
Loop

Tim0_isr:
  Timer0 = 100    '10ms
  D = Getadc(2)
  Du = U / 40
  U = U - Du

```

```

U = U + D
Ticks = Ticks + 1
U1 = D - Du
If U1 > 0 And U2 < 0 Then
  T = Ticks * 10
  F = 60000 / T
  Ticks = 0
  If F > 45 Then
    Print T
    Print F
    Locate 1 , 1
    Lcd T
    Lcd " ms "
    Locate 2 , 1
    Lcd F
    Lcd " /min "
  End If
End If
U2 = U1
Return

```

Listing 5. Encoder evaluation in Arduino C.

```
//Encoder A0/A1 PWM1, A4/A5 PWM2
...

int clk1 = A0;
int dt1 = A1;
int clk2 = A4;
int dt2 = A5;
int pwm1 = 9;
int pwm2 = 10;

...

void loop() {
  new1 = digitalRead(clk1);
  if((new1==0) & (old1==1)){
    if (digitalRead(dt1)==0) d1++; else d1--;
    Serial.println (d1);
    if (d1 > 250) d1 = 250;
    if (d1 < 0) d1 = 0;

    analogWrite (pwm1, d1);
    lcd.setCursor(0, 0);
    lcd.print(d1 * 20);
    lcd.print(" mV ");
  }
  old1=new1;

  new2 = digitalRead(clk2);
  if((new2==0) & (old2==1)){
    if (digitalRead(dt2)==0) d2++; else d2--;
    Serial.println (d2);
    if (d2 > 250) d2 = 250;
    if (d2 < 0) d2 = 0;
    analogWrite (pwm2, d2);
    lcd.setCursor(0, 1);
    lcd.print(d2 * 20);
    lcd.print(" mV ");
  }
  old2=new2;
}
```

Measuring pulses with Bascom

In Bascom you have to sort out time measurement on your own because there is no permanent timekeeping process running in the background, only whatever you arrange yourself. A Timer Interrupt works well for this. Here we are using Timer 0 for the timekeeping. The Interrupt routine is invoked every 10 ms, meaning that you can assess time to this level of resolution. With Interrupts comes precision. In this case it means that the voltage measurements are also carried out by the Interrupt. This has the advantage that they take place in extremely accurate 10 ms tempo, eliminating any potential AC line frequency (50 or 60 Hz) interference.

For an ATmega 10 ms represent a long time and that means the entire analysis can be carried out in the (Listing 4). As in the Arduino model above, this involves generating the moving average, identifying the pulse edges (flanks), eliminating mea-

surement errors caused by missing pulses and outputting to the Terminal and the LCD. The end product is a Bascom pulse meter with extremely similar characteristics to those of the Arduino project.

Anyone who has battled with a computer mouse of the older design will have encountered the type of light barrier used for measuring rotation. The crucial factor is that you can detect not only impulses but also (using two light barriers) the direction of rotation. The two light barriers are arranged so that rotation produces squarewave pulses offset in phase by 90 degrees. The same signals are produced by the rotary encoder in the Sensor Kit (Figure 9). However, instead of using light barriers, the encoder employs mechanical contacts. It also has clearly detectable mechanical detents, enabling you to feel, track and count each step made. A complete rotation covers 20 steps by the way. Twenty may not sound plentiful but since there

Listing 6. Encoder evaluation in Bascom.

```
'Encoder.BAS C0/C1 PWM1a, C4/C5 PWM1b
...

Do
  New1 = Pinc.0
  If New1 = 0 And Old1 = 1 Then
    If Pinc.1 = 0 Then D1 = D1 + 1 Else D1 = D1 -1
    If D1 > 1023 Then D1 = 1023
    If D1 < 0 Then D1 = 0
    Pwm1a = D1
    Print D1
    Locate 1 , 1
    Lcd D1

    End If

    Old1 = New1

    New2 = Pinc.4
    If New2 = 0 And Old2 = 1 Then
      If Pinc.5 = 0 Then D2 = D2 + 1 Else D2 = D2 -1
      If D2 > 1023 Then D2 = 1023
      If D2 < 0 Then D2 = 0
      Pwm1b = D2
      Print D2
      Locate 2 , 1
      Lcd D2
    End If
    Old2 = New2
  Loop
```

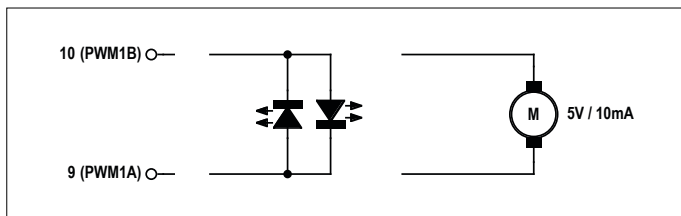



Figure 11. Bridge output stages.

is no physical end-stop, you could select, say, hundreds or thousands of steps, which would be very practical for setting a frequency or voltage precisely. A pushbutton switch is also included, although we do not use this here.

Although both contacts are in fact equivalent, one of them is designated the Clock Pin (CLK) and the other is the Data Pin (DT). Figure 10 shows the output signals for one rotation. Evaluating them is really easy: you wait for a trailing edge at CLK and then examine DT in order to decide whether to increment or decrement the counter status accordingly.

The sample program (Listing 5) employs two encoders, enabling us to have two analog output signals PWM1 and PWM2 on Arduino outputs 9 and 10. With an additional lowpass filter you could then have a dual adjustable voltage source for all-purpose measurement use. You could also use this for creating color mixes using the red/green LED in the Sensor Kit with absolute sensitivity and reproducibility. The preset values can additionally be output in serial format and indicated on the LCD. The rotary encoder includes its own pullup resistors. At the same time internal pullups are enabled on the Arduino. So using the same program you could use either one or two encoders. The first is connected to inputs A0 and A1. Purely by coincidence we have here pushbutton switches S1 and S2 on the Extension Shield. Because all contacts are at GND potential no conflict arises. But in an emergency you could transpose the first output value d1 with the pushbuttons. S1 would then provide the Clock signal and S2 the direction of rotation.

We connect the second encoder to A4 and A5. On the Arduino the corresponding Portpins are brought out twice, the second time diagonally opposite as SDA and SCL of the I2C interface. Nearby there is also a GND Pin. What is missing, however, is an additional 5 V connection, which we need for the pullups on the encoder. No problem, however, as we switch Pin 13 HIGH, to which the Arduino LED is also connected. This is always a solution: whenever you are short of a GND or VCC connection, just program a Portpin for this purpose.

Encoding in Bascom

The Bascom version of the program is constructed almost identically. You can switch between languages by mutually accepting the source code, retaining the Variable names, and revising only the specific syntax properties. In the end result there is only one real difference: in Bascom you normally use the two PWM outputs of Timer1 with 10-bit resolution, meaning you now have 1023 steps and a step width of around 5 mV. Two precisely adjustable PWM outputs, which almost scream out for an experiment on the theme of bridge output stages, especially since the two PWM signals originate from the same timer and have precisely matching pulses with synchronized leading edges. A green and a red LED can be connected in

anti-parallel between the two outputs (Figure 11). If you provide a middle-ranging voltage, like 2500 mV for example, both LEDs remain dark. If one channel is out of balance, one of the LEDs starts to light up.

Likewise one could connect an extremely frugal (low current) DC motor (cassette recorder type, starting current 10 mA) and then control the direction and rotational speed. Yes, it's true: actually you should not do this without a proper motor driver circuit. But it does work in fact. There is no danger of induced voltages in this case, because the outputs are always in a low-impedance state. Except perhaps when you reset the Arduino during operation, because then the connections are high-impedance. So you are better off making only brief tests and removing the motor again for as long as the program is still running... ◀

(160302)

Web Links

- [1] www.elektor.com/arduino-sensor-kit
- [2] www.elektormagazine.com/160152
- [3] www.elektormagazine.com/160173
- [4] www.elektormagazine.com/160210
- [5] www.elektormagazine.com/160302
- [6] www.elektormagazine.com/140009

Advertisement

Mercury Coulometers

Peculiar Parts, the series

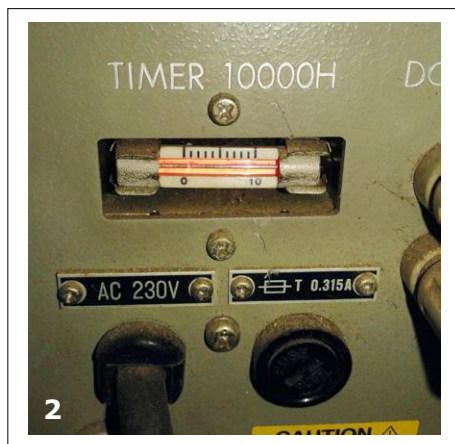
By **Neil Gruending** (Canada)

Coulometers are almost everywhere, quietly measuring the coulombs going in and out of battery packs and estimating the remaining battery capacity. Like most modern solutions they are implemented as an integrated circuit which why I was intrigued when I read a thread on the 'TekScopes' mailing list about a

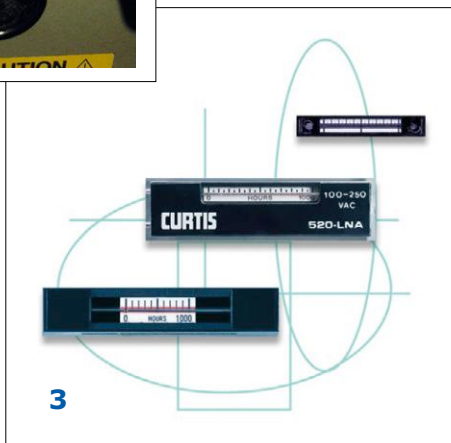
side of the gap to be electrochemically transferred to the cathode side which made the gap to move towards the anode. Reversing the current will also reverse the gap. The rate of movement is proportional to the amount of current and the distance moved shows the current over time. This behavior is what makes

still 20 of them still on the moon in the descent module.

Mercury coulometers do have some drawbacks though. The obvious one is that a large impact or shock can damage the capillary tube or disturb the electrolyte solution in the gap which can prevent the meter from working properly, espe-

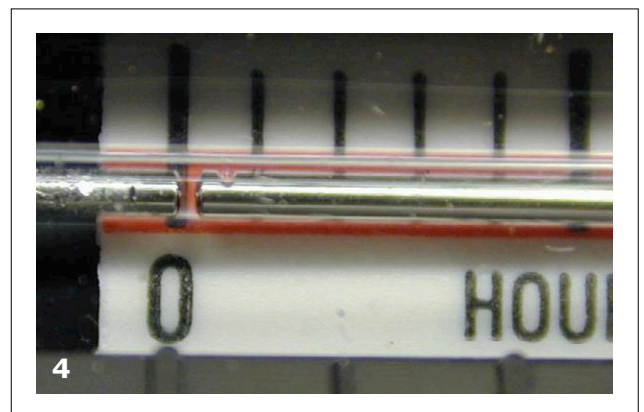
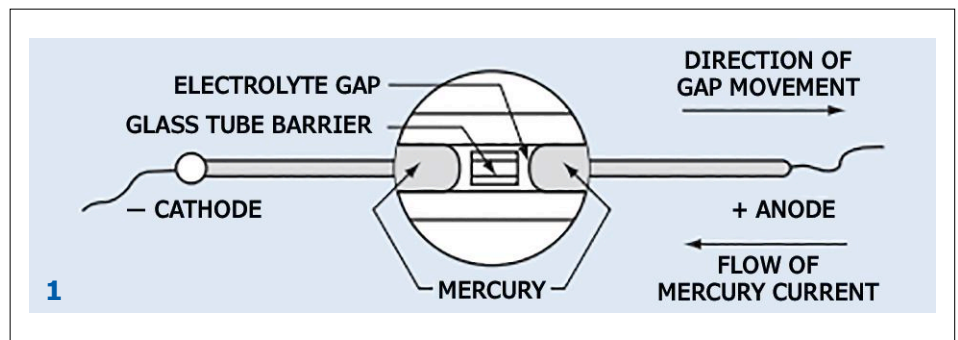


mercury coulometer that was being used as an elapsed time counter to estimate the amount of time that a piece of test equipment was on. You would never know they were there unless you looked inside so let's take a closer look at these impressive devices.



it a coulomb meter and also makes them suitable as an elapsed time counter.

The simple design of the coulometer means that it is very lightweight and very shock resistant. It also makes them very flexible because the total number of counted coulombs is fixed so the current defined the total number of hours counted which was usually in the 1000's of hours. These properties meant that mercury coulometers were used in some interesting places like timers in the Apollo Lunar Landing module. In fact, there are



cially if the mercury leaks out. Another issue is that if the solution gets to the end of the tube then it can dissolve the electrode and cause a leak.

No one manufactures mercury coulometers in North America anymore so unless you find one in a piece of equipment somewhere (**Figures 2, 3, 4**) they are hard to find. However, it is possible to find Russian versions available occasionally on EBay if you would like to get your hands on one — just be careful with the mercury. ◀

(160178)

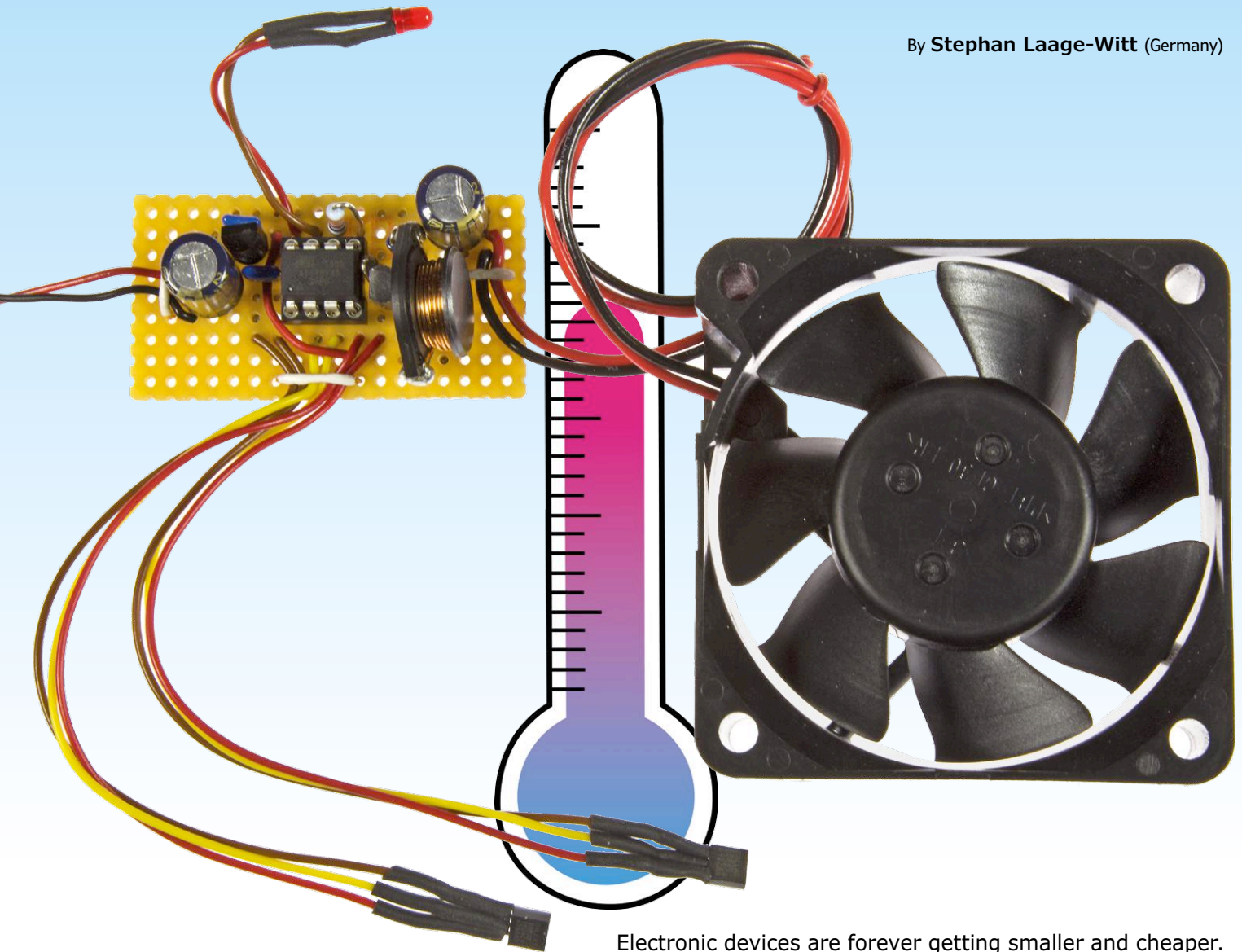
Image sources

(1) globalepower; (2) AskElectronics; (3) Octopart; (4) 4hv.

Keep Cool with an ATtiny

Temperature-dependent fan control

By **Stephan Laage-Witt** (Germany)



Electronic devices are forever getting smaller and cheaper. One of the first casualties of such progress is a proper, well-thought-out cooling system: fans either run continuously or have a simple two-point (on/off) control mechanism. However, designing a small, effective, active cooling system is not exactly black magic.

Electronic devices often operate at well below their maximum load for most of the time, and as a result a continuously-operating fan is considerably more of a nuisance than it needs to be. In my small workshop, for example, I have two such sources of irritating background noise:

an inverter that provides my mini-office with 230 V from my 12 V solar installation, and a rather old digital oscilloscope. Both units work well, but nevertheless are continuously whirring away. As an engineer, this naturally prompted me to turn my hand to finding a simple

and low-cost, but precise and effective way to modify the devices to solve the problem. With just a few components it is possible to add a simple controller to an off-the-shelf 12 V DC fan that is so small that it can easily be fitted into almost any equipment. The project has a wide

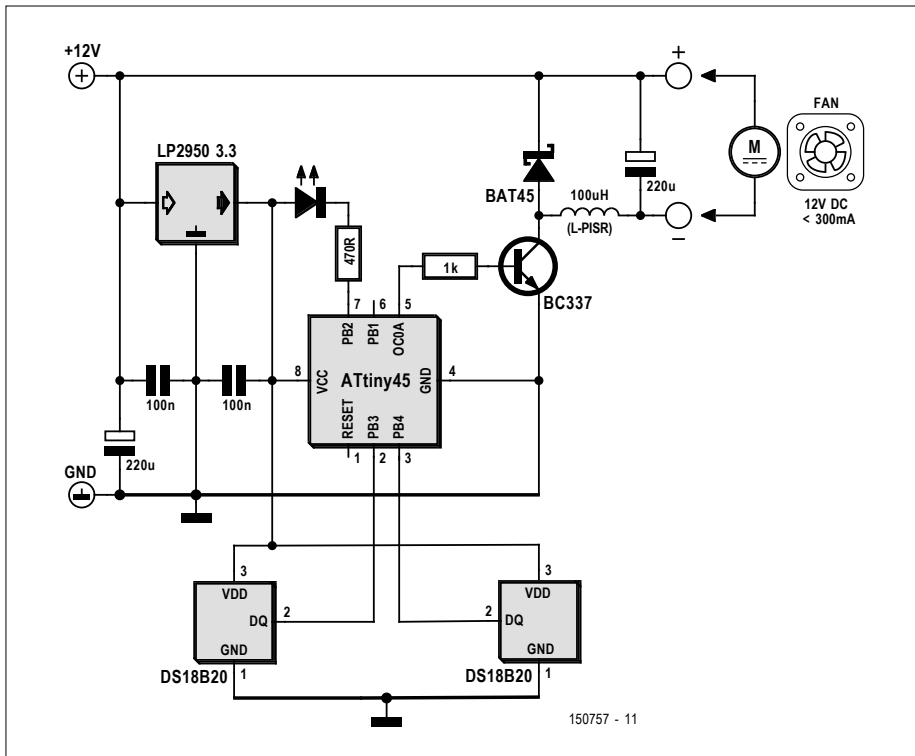


Figure 1. Circuit diagram of the fan controller.

range of uses and can even be built into your own projects where active cooling is required. Assembly is straightforward, ideal for a rainy Sunday evening.

ATtiny keeps things cool

Our starting point is, as you might expect, an AVR microcontroller. In this case we use an ATtiny45. With the aid of two type DS18B20 digital temperature

sensors the microcontroller can monitor the temperature at two different places within the device. The sensors themselves do not require calibration and come with a guaranteed worst-case accuracy of 0.5 °C, which is more than good enough for our application. Over the desired temperature range the microcontroller generates a pulse-width modulated (PWM) signal that is used to provide continuous

control to a fan.

Between the microcontroller and the fan there is a discrete step-down converter to supply variable power to the fan consisting of a transistor, a Schottky diode and a coil. The smoothed voltage across the output capacitor is suitable for driving conventional brushed motors as well as brushless DC motors.

The other components are a small linear voltage regulator to provide power to the AT tiny, and LED warning indicator, a few capacitors and two resistors. And that is all there is to the hardware.

The circuit diagram in **Figure 1** shows how the components work together. The temperature sensors are read over port pins PB3 and PB4. The pull-up resistors required by the sensors are not shown in the circuit, as the pull-ups built in to the microcontroller's input ports are used instead. Port pin PB0 outputs the PWM signal OC0A, which drives the BC337.

This transistor can supply a current of up to 300 mA to the fan, which in my applications was more than enough. If you wish to drive larger fans, a more powerful transistor and a lower base drive resistor can be used. The coil (an L-PISR SMD power inductor available from Reichelt), the Schottky diode and the low-ESR electrolytic capacitor produce the supply voltage for the fan: they form a step-down converter referenced to the positive supply.

Apart from the coil, leaded components can be used throughout, and the whole thing can quickly be assembled on a small piece of prototyping board.

Software

Of course, the ATtiny needs some software in order for it to do its job. The firmware must be programmed into the device using a suitable external programming adapter before it is fitted to the board. It is therefore a good idea to use a socket for the microcontroller so that if necessary it can be removed to have its software update at a later date if necessary.

The software is written in C and is thoroughly commented, and it can be downloaded for free from the web pages accompanying this project [1]. The ATtiny is clocked internally at 8 MHz: as delivered from the factory the devices normally have their CKDIV8 configuration fuse set, and for this application it must be cleared.

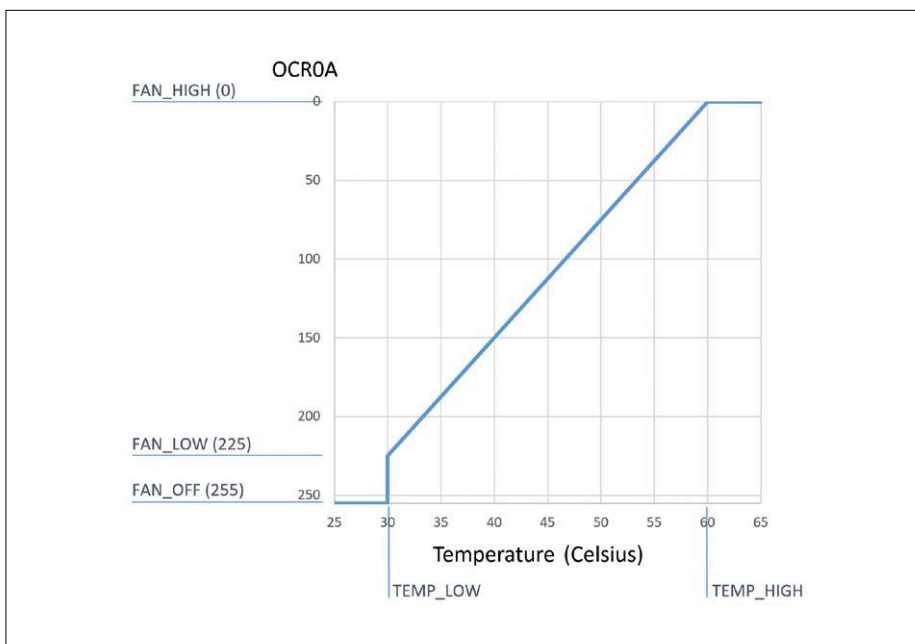


Figure 2. Relationship between OCR0A value and measured temperature.

The configuration parameters can be found in the '#define' statements at the beginning of the program. The temperature range and the PWM threshold values can be altered here as desired.

The largest part of the software consists of the interface routines for the temperature sensors, which use the 1-wire protocol. Accurate timing is essential, and this is provided by the delay routines in the GCC library. The DS18B20 offers a range of accuracies (resolutions from 9 to 12 bits). In this case we use the lowest resolution, 9 bits, which gives a smallest temperature step size of 0.5 °C as mentioned above. The functions in the program return the temperature values in Celsius as binary numbers. The protocol used by the DS18B20 and corresponding AVR routines are described by Gerard Marull Paretas in his fine application note [2]. The ATtiny45 has two eight-bit timers. Timer0 is used for the PWM signal, and is configured in fast PWM mode. The resulting PWM frequency is 31.25 kHz, which is ideal for this application. Register OCR0A determines the duty cycle of the output. Timer1 is used to generate a periodic interrupt at 2 Hz. Alternate interrupts cause the main code to carry out its processing; when this work in the main loop is done the microcontroller is put into sleep mode to save power.

The greater of the two temperature readings is taken and used to determine the correct duty cycle for the PWM output. **Figure 2** shows the relationship between temperature value and OCR0A register contents. Note that the OCR0A value is inversely related to the voltage across the fan: higher register values give lower voltages and vice versa. Setting the register to 255 switches the fan off completely (FAN_OFF). The fan has a minimum start-up voltage, and this is specified in terms of the OCR0A value as FAN_LOW (225 in this example). When starting the fan, that is when switching from FAN_OFF to FAN_LOW, the fan is run briefly at full power in order to ensure that it starts moving. The length of the full-power pulse is 30 ms (FAN_STARTUP_DELAY). Both FAN_LOW and FAN_STARTUP_DELAY need to be adapted to suit the particular fan in use to ensure reliable operation at start-up and low speeds. If in doubt, use a smaller FAN_LOW value and/or a longer start up delay to be on the safe side.

The red LED indicates a fault condition. The error flag is set if one of the two temperature sensors does not respond to its reset signal within the time specified in its datasheet. The LED will then blink at 1 Hz to indicate that there is something or other wrong with the temperature sensors, such as a wiring fault. The LED's second function is as an overtemperature indicator: if the upper range of the temperature control range is exceeded the LED will light steadily.

Installation

The circuit board is so small that space can be found for it inside practically any equipment (see **Figure 3**). It is simply connected between the equipment's own power supply and the fan. The controller circuit itself draws only about 2 mA. For a first test connect the fan in the equipment to the circuit, and power the circuit from an external 12 V mains supply. When power is applied the fan should run for 2 s (adjustable using POWER_ON_DELAY) on full power, and the LED should light during this period. This provides some indication that everything is working. Then the fan should switch off and the LED should extinguish. Now take one of the temperature sensors between your fingers to warm it up (or bring it near to the tip of your soldering iron). After a while the start-up threshold of 30 °C will be reached and the fan will start to run.

The other sensor should behave in the same way. Next you can test that the fan starts up reliably and that it runs stably at low rotational speeds: if not, you will need to adjust the values of FAN_LOW and FAN_STARTUP_DELAY. Once everything is working the board can be mounted in a suitable position in the equipment and connected to the internal power supply. The temperature sensors should be mounted close to the sources of heat. In

my case I used a drop of superglue to hold them in place on the heatsinks of some power transistors. And with that, the project is complete. I have been using two of these tiny controllers continuously for several months.

The fans run seldom and usually slowly, as the equipment is usually only running at low power, and the noise level is enormously reduced. Also, the controller helps give peace of mind in that the temperature in the equipment's enclosure is always kept within reasonable limits. ◀
(150757)

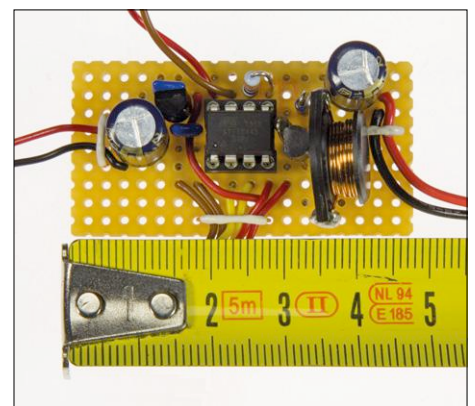


Figure 3. The assembled fan controller board.



Web Links

- [1] Project pages: <http://www.elektormagazine.com/150757>
- [2] Using the DS18B20 digital temperature sensor on AVR microcontrollers. Gerard Marull Paretas, September 2007: http://teslabs.com/openplayer/docs/docs/other/ds18b20_pre1.pdf

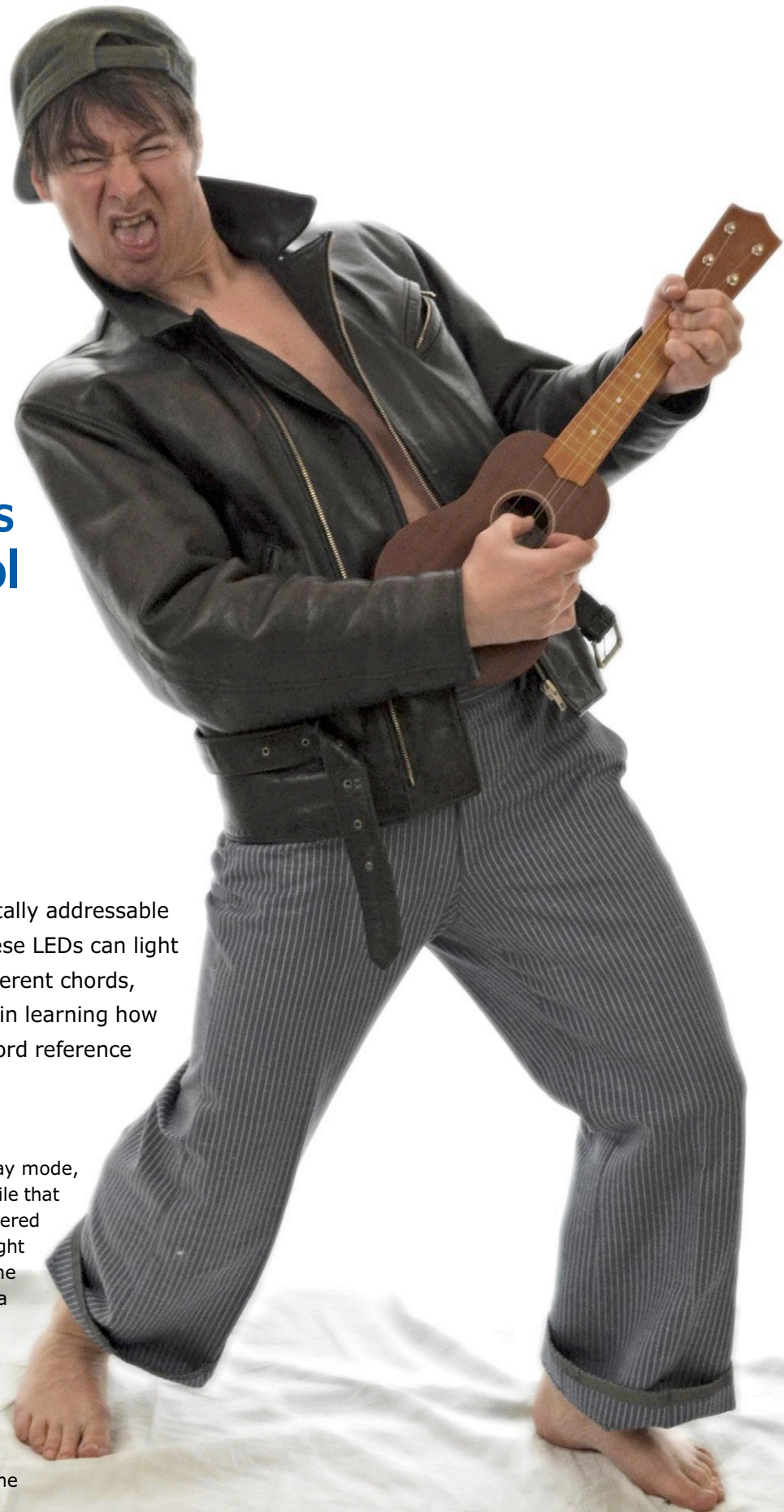
Ukule-LED

serially addressable LEDs as ukulele learning tool

By **Raghav Ayurkudi Subramaniam** (USA)

Ukule-LED is a modified ukulele with digitally addressable RGB LEDs embedded in its fretboard. These LEDs can light up in configurations corresponding to different chords, which can aid a beginning ukulele player in learning how to play certain chords, or can act as a chord reference for a more advanced player.

Ukule-LED has two modes of operations. In play mode, the user can feed the system a song file, a text file that contains the tempo, time signature, and an ordered listing of the chords in a song. The ukulele will light up the correct chords at the correct times in the song. In practice mode, the user can specify a single chord, which is lit up indefinitely. We use sixteen RGB LEDs, embedded in the fretboard before the first four frets on each string. These LEDs are wired to an Atmel microcontroller, which is mounted on the ukulele itself. The ukulele can be connected to a PC, which runs a companion program with a command line interface.



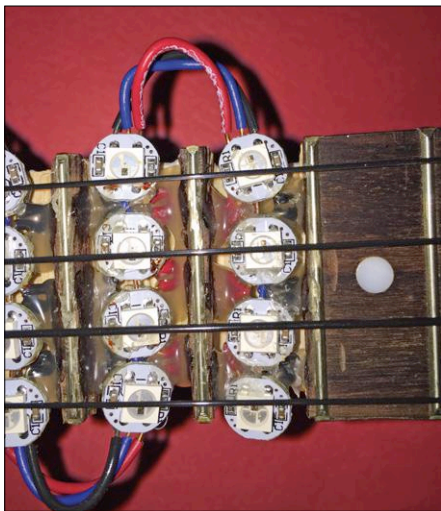


Figure 1. This is a closeup of the completed ukule-LED fretboard. The 16 RGB LEDs are connected in series and connected to a single microcontroller I/O pin.

Features

- RGB-LED-driven music learning system
- Play and Practice modes
- Easy-to-understand file format

Hardware

We decided to use Atmel's ATmega1284P microcontroller for this project because it has the two 16-bit timers we needed for our project. One was used for task scheduling, and the other was used to keep time for the music. Additionally, we used Bruce Land's custom PCB for the ATmega1284P [2], which includes an external crystal, an LED, and a power regulator, among other components. This made interfacing with the microcontroller easy. The Elektor Platino board can also be used with the ATmega1284P.

To prepare the ukulele, we used a Dremel tool to cut grooves in the ukulele's fretboard to accommodate the LEDs. Critically, we had to keep all the fret wires intact and keep the LEDs below them in order to maintain playability of the ukulele. Otherwise, the LEDs, not the fret wires, would stop the string, pulling notes out of tune. We hot glued four sets of four LEDs to the grooves in the fretboard, one for each of the first four frets on each of the four strings. **Figure 1** shows the completed ukulele fretboard. We used Adafruit's NeoPixel Mini PCBs, each of which includes a WS2812B RGB LED. All of the RGB LEDs are driven from a single microcontroller I/O pin.

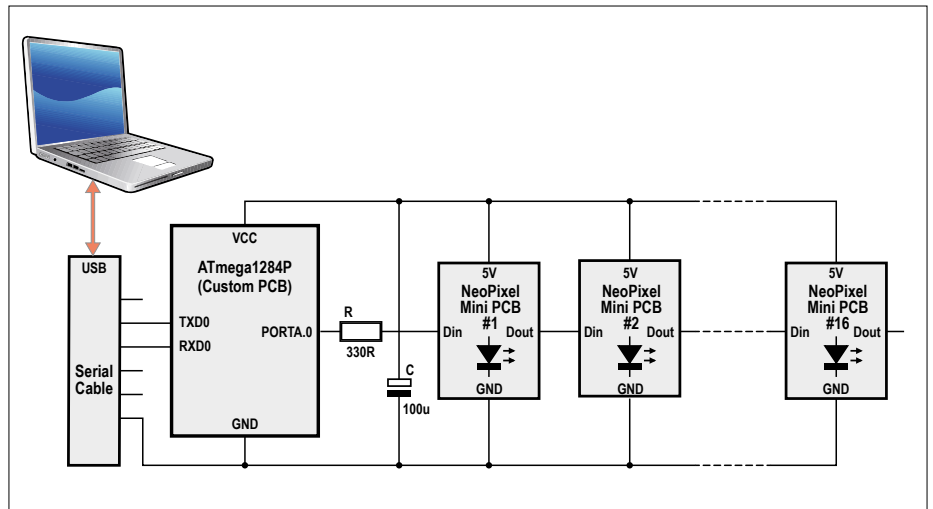


Figure 2. This is a schematic of all of the ukule-LED hardware. This includes the microcontroller, the RGB LEDs, and the serial connection to the user's PC.

WS2812Bs are driven serially, using a very timing-sensitive 1-Wire protocol, with the data output, power, and ground pins of one connected to the data input, power, and ground pin of the next. The data input pin of the first LED is connected to an I/O pin on the microcontroller through a 330 Ω resistor to protect the first LED from any voltage spikes. As an additional protection, there is a 100 μ F capacitor between the power and ground lines to prevent large inrush currents from destroying the first LED (according to Adafruit's NeoPixel PCB documentation [3]).

Soldering the RGB LEDs together was the most time-consuming part of building ukule-LED. Each of the sixteen NeoPixel PCBs needed to be individually soldered. We used very short lengths of wire, soldered to the pads on the PCBs, such that

there was about 1-2 millimeters of space between each of the four PCBs placed in each groove in the ukulele's fretboard. Next, we wired these four groups using longer lengths of wire. Finally, we hooked all 16 NeoPixel PCBs to the main solder board with three long lengths of wire. Our next task was adequately powering the LEDs. Because we only powered four LEDs at a time and did not run them at maximum intensity, we could drive the RGB LEDs using the microcontroller's VCC pin instead of resorting to an external power source. In the end, we only needed to use a 9 V power supply to power the entire system. A schematic of the electronics involved in ukule-LED is shown in **Figure 2**.

Finally, we connected all of the electrical components — the microcontroller, the LED circuitry, the power circuitry, and

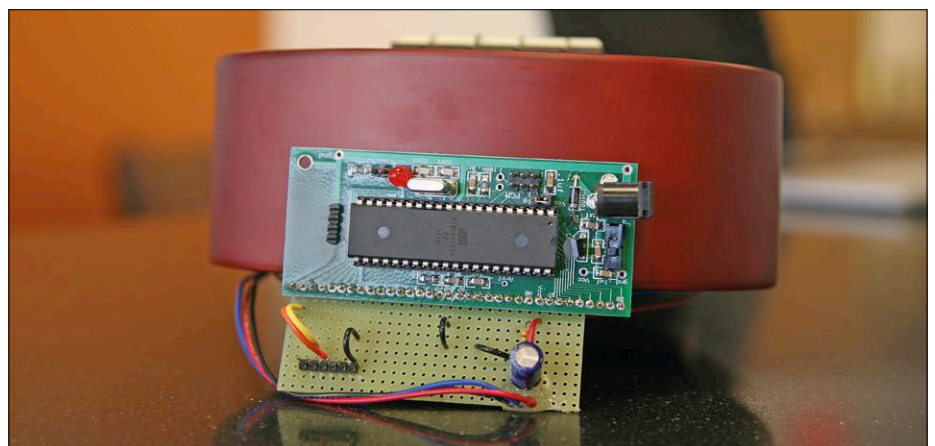


Figure 3. This is the solder board, taped to the bottom of the ukulele. It contains all of the auxiliary components used to drive the LEDs and the serial communication.

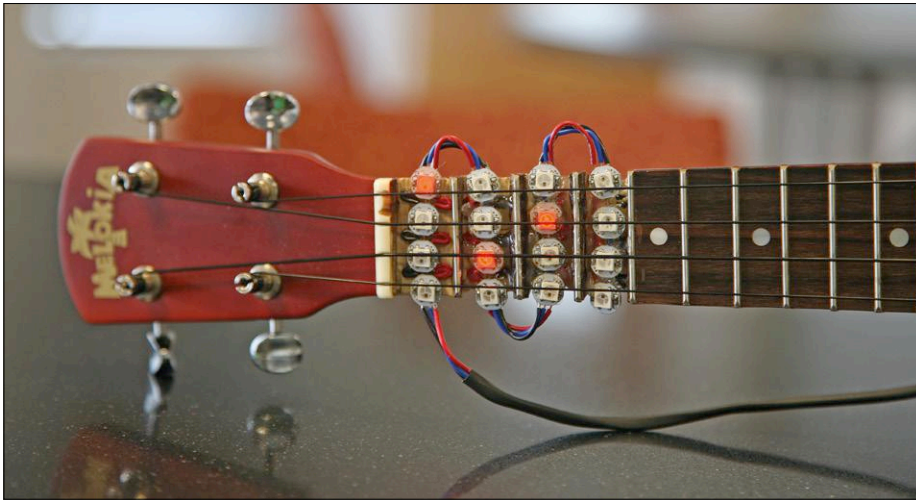


Figure 4. Here is a Gm chord. As a minor chord, it is lit up in red.

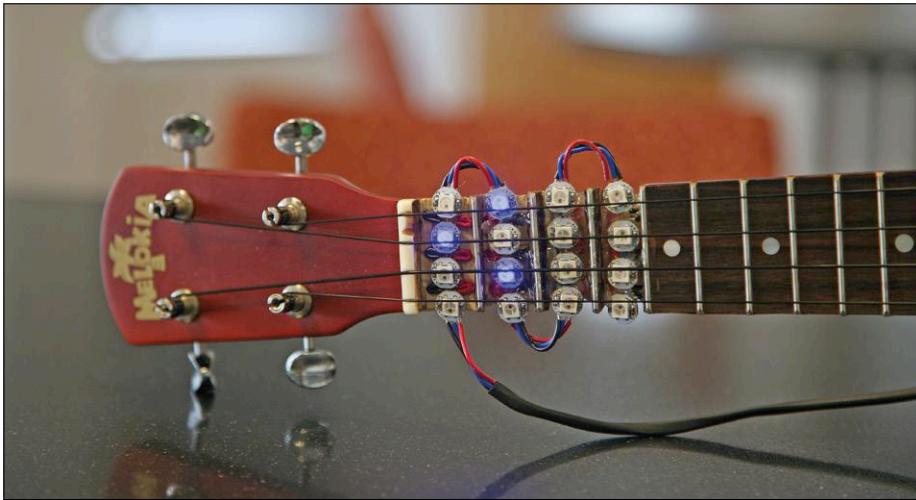


Figure 5. Here is a G7 chord. As a seventh chord, it is lit up in blue.

some wiring and pins for a serial-to-USB cable — to a small solder board which we then glued to the base of the ukulele, as shown in **Figure 3**.

Command line interface

To interact with the microcontroller, our project requires a computer running a Python-based command line interface (CLI). We used docopt [4], a command line argument parser for Python. There are two command formats that a user can use to start a ukule-LED session:

```
ukule_led practice "chord" port "portname"
ukule_led play "file" port
"portname"
```

The first type of command tells the microcontroller to go into practice mode. The

arguments for this mode are **chord**, which is the chord to be played—A, C#m, or Gb7, for example, and the other argument is **portname**, which is the name of the serial port on the computer which is connected via the serial-to-USB cable to the microcontroller. The second command type tells the microcontroller to go into play mode. The arguments are **file** which specifies the path to the song file on the user's computer, and **portname**, which serves the same function as previously described. We designed our own song file format for ukule-LED. We felt existing standards and protocols, such as MIDI, were overkill. Below is an example of a song file.

```
240 4
|A B C D|E F G -|A - - B|
|A - Bbm A7|Bm X A#7 -|A - X D|
```

The first line of the song file contains two numbers, the tempo of the song in beats per minute, and the time signature of the song, in beats per measure. Only one chord can be played per beat. The remainder of the song file contains the measures of the song, delimited by '|' characters.

We support major, minor, and seventh chords. We designed ukule-LED to light up major chords in green, minor chords in red, and seventh chords in blue. Examples of chords are shown in **Figures 4 and 5**. A chord's base can be any note, with '#' denoting a sharp and 'b' denoting a flat. Finally, a dash '-' tells the system to hold a previous chord for another beat, and an 'X' means no chord should be displayed during that beat.

When the Python CLI receives a command from the user, it uses docopt to parse the command's arguments. The command is then converted into a string so that it can be serialized and sent over the serial connection. We developed an encoding for chords that assigns each chord name a unique chord number starting from 1. Additionally, we use 0 to denote an 'X' in the song file. An example of a practice mode string is "1/17", and an example string for play mode is "2/200/4/14/13/02/30/-1". The first number in the string is a flag; '1' denotes practice mode and '2' denotes play mode. In practice mode, the number following the '1' denotes the chord to be played. In play mode, after the '2', the next number provided is the tempo, and the third is the number of beats per measure, which are 200 and 4 in the example respectively. The next numbers represent the chords of the song in sequential order. There can be indefinitely many of them, and the list is terminated by a -1. Using Chris Liechti's pySerial package [5], the program then opens a connection to a serial port provided as a command line argument by the user, and writes the string to the port.

Embedded software

The microcontroller runs two major tasks. The first task receives commands from the CLI via a serial port. The microcontroller stores in its memory a chord array whose indices are chord numbers and whose values denote the configuration of LEDs to be lit up. Every hundredth of a second, a second task polls the value of a pointer into this chord array and updates lights up the correct LEDs. In practice

mode, this pointer remains constant. However, in play mode, a timer with frequency specified by the tempo in the song file updates the pointer according to the chords in the song file (**Figure 6**). We used the TinyRealTime (TRT) kernel [6], developed by Dan Henriksson and Anton Cervin, to handle the scheduling of these two tasks. We also used Alan Burlison's WS2812 LED driver [7]. This driver is written in assembly and takes care of the timing needs of the RGB LEDs. By controlling which LEDs are turned on, and in what color, we can control which chords are displayed.

The embedded code contains a main function that first sets up the LED driver, specifying the I/O port that the LEDs are physically connected to. Next, it sets up TRT according to the TRT documentation. TRT uses timer1, one of the ATmega1284P's two 16-bit timers. `main` next sets up the other 16-bit timer, timer3, which drives the timed transitions between chords.

The first real-time task, carried out by the function `serialComm`, is responsible for waiting for user input to arrive at the serial port, and parsing it when it does come in. It allocates a command buffer `cmd` of size 1,000 bytes, long enough to accommodate a majority of songs, and calls the TRT function `gets(cmd)`, which waits on a semaphore that is signaled when TRT receives input from the serial port. When the function stops waiting, a command (an array of numbers extracted from the string transmitted from the CLI via serial) will have been buffered in `cmd`. If the command is a practice mode command, `serialComm` sets the variable `chord_idx` to the number corresponding to the single chord to light up. `chord_idx` is the pointer into the chord array. The value of `chord_idx` remains constant until another command is received. If the command is a play mode command, `serialComm` reads the second number in the command, which is the tempo in beats per minute of the song. It takes this tempo value and converts it into the timer3 period according to the following formula.

$$\text{period} = (62500 \times 60) / \text{tempo} - 1$$

After the timer3 period is set, the function extracts the third number in the command, which is the number of beats per measure. Finally, `serialComm` calls a helper function that walks through

Acknowledgements

I'm grateful for the support of Dr. Bruce Land and my TA, Eileen Liu. Ukule-LED was almost painless thanks to the open-source software written by Dan Henriksson, Anton Cervin, Jeorg Wunsch, Alan Burlison, Chris Liechti, and the developers behind docopt. Finally, this project would have been impossible without my lab partner, Jeff Tian.

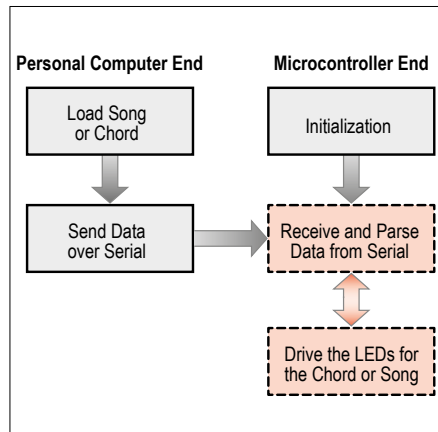


Figure 6. This flowchart shows the different software tasks performed by ukule-LED. This includes two real-time tasks on the microcontroller and a command-line interface running on the user's PC.

the `cmd` array, converts the numbers from characters to integers, and inserts them into a global array called `song`. We use the timer3 ISR to drive the song. Every time the ISR fires we run `chord_idx = song[song_idx]`, where `song_idx` is the current index into `song`, and then increment `song_idx`.

The second real-time task simply polls the `chord_idx` value set by the ISR, gets the corresponding array from `lookup_table`, and feeds it to the LED driver.

Conclusions

My goal in putting together ukule-LED was to build a usable and extensible system for learning how to play the ukulele.

It is similar in weight and feeling to an unmodified ukulele. The software interface is simple and easy to use, thanks to docopt. ukule-LED can easily be extended by editing the files `chord_map.py` and `chord_lookup_table.h`, which determine what chords are supported and what colors individual chords light up in. In one possible extension, by adding to both files, a user can add support for all types of chords. ◀

(160439)

Web Links

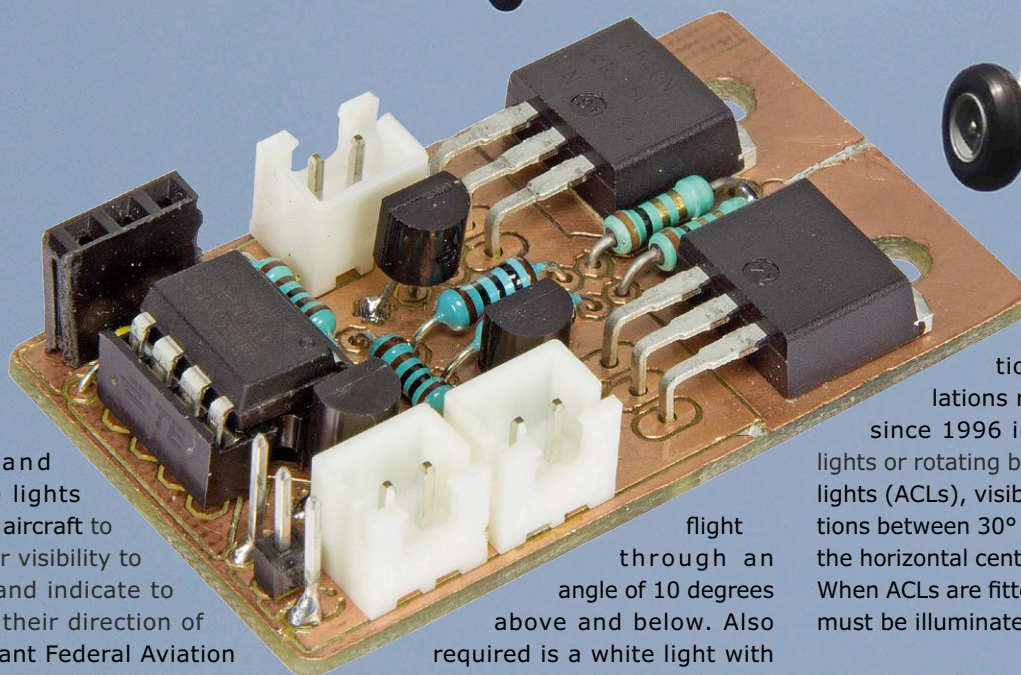
- [1] www.elektormagazine.com/160439
- [2] <http://people.ece.cornell.edu/land/PROJECTS/ProtoBoard476/>
- [3] <https://learn.adafruit.com/adafruit-neopixel-uberguide>
- [4] <http://docopt.org/>
- [5] <http://pyserial.sourceforge.net/>
- [6] www.control.lth.se/~anton/tinyrealtime/
- [7] http://bleaklow.com/2012/12/02/driving_the_ws2811_at_800khz_with_a_16mhz_avr.html

Making Lights Meaningful

For all model aircraft

Flying model aircraft should duplicate their full-size prototypes right down to minor details, so there's no reason for external lighting to be an exception. Adding these lights is a minor task and as a bonus, they can also provide a visual indication of the battery level onboard the models.

By **Udo Hunzelmann**, DK8VT@DARC.de (Germany)



Throughout the world air traffic licensing regulations stipulate and standardize lights displayed by aircraft to improve their visibility to other craft and indicate to other pilots their direction of flight. Relevant Federal Aviation Administration (USA), German and Wikipedia information can be found here [1]. In essence, aircraft must carry colored lights (red on the left, green to the right) that can be seen in the direction of

flight through an angle of 10 degrees above and below. Also required is a white light with unobstructed visibility from right and left over an angle of 70 degrees upwards and downwards, usually attached at the rear. These lights may be permanently illuminated or flashing.

As well as these navigation or position lights, the regulations require on craft built since 1996 intermittent (strobe lights or rotating beacons) anti-collision lights (ACLs), visible ideally in all directions between 30° above and 30° below the horizontal center line of the aircraft. When ACLs are fitted, the position lights must be illuminated constantly.

Making lights deliver extra information

What the regulations prescribe for real airplanes (and helicopters) are not expensive to provide on model aircraft.

As small-scale pilots we do not need to follow those legal requirements to the letter, which leaves us free to enhance these external lights with a small but highly useful function: a battery level indication visible from afar!

A constant problem when flying models is that you have no idea how full or exhausted their batteries are. To remedy this problem, I have developed a small flasher circuit, which modifies the rate of flashing according to the battery charge state. In addition, a buzzer or sounder can be coupled to the circuit that reports the location of the model audibly.

The circuitry in **Figure 1**, which is linked to the servo output of the model's flight controller by connector K1, makes use of a small ATtiny45 microcontroller with just eight pins that control three

ment (only some software) I constructed a kind of window comparator for $V_{CC}/2$:

- $V_{CC}/2 > 3.7\text{ V}$ Battery is fully charged: 3x flashes
- $3.4\text{ V} \leq V_{CC}/2 \leq 3.7\text{ V}$ Battery is 50% charged: 2x flashes
- $V_{CC}/2 < 3.4\text{ V}$ Battery is flat (discharged): 1x flash

This ploy does have a minor drawback in that it assumes that the (first) cell measured is representative of all the others. We would need to develop further circuitry to ensure the total battery voltage (and each of the cells) was measured. Building this circuit on its PCB, which is available from the Elektor Store (you can also download its layout from the project web page [3]), takes just a short

switched outputs with N-channel MOSFETs. Although the buzzer connected to K2 draws very little current (in the region of 5 mA at this low operating voltage according to type), making a BS170 small-signal switching transistor fully adequate, the MOSFETs for the two high-power LEDs of the strobe lights connected to K3 and K4 are another matter. These two IRF540 devices (in TO220 packages) handle high currents flowing through the LEDs. The subsidiary circuits — with R3 and R6 acting as shunts plus a pair of BC547C bipolar transistors — provide simple current limiters and ensure that the MOSFET stages are deactivated immediately in the event of a short circuit involving the LEDs.

The circuit is powered by the battery pack (using at least two LiPo cells) via its balancer connection cable [2]. A three-way pinheader (K5) is provided for this on the small PCB in **Figure 2**. The high-intensity LEDs use the full voltage available (V_{CC}) of 7.4 V, whereas the controller and the buzzer use only half ($V_{CC}/2$), namely 3.7 V.

The ATtiny45 has a remarkable internal function: it can make an analog measurement of the voltage on its V_{CC} pin! Naturally I have made use of this feature and without employing a single extra compo-

60 minutes or so. No particular difficulties are expected, especially as only through-

PROJECT INFORMATION

	model aircraft	RC
	position lights	
	ACL	

	entry level
	intermediate level
	expert level

	1 hour approx.
--	----------------

	Soldering iron, Ttiny45 programmer, Arduino-IDE
--	---

	€20 / £17 / \$21 approx.
--	--------------------------

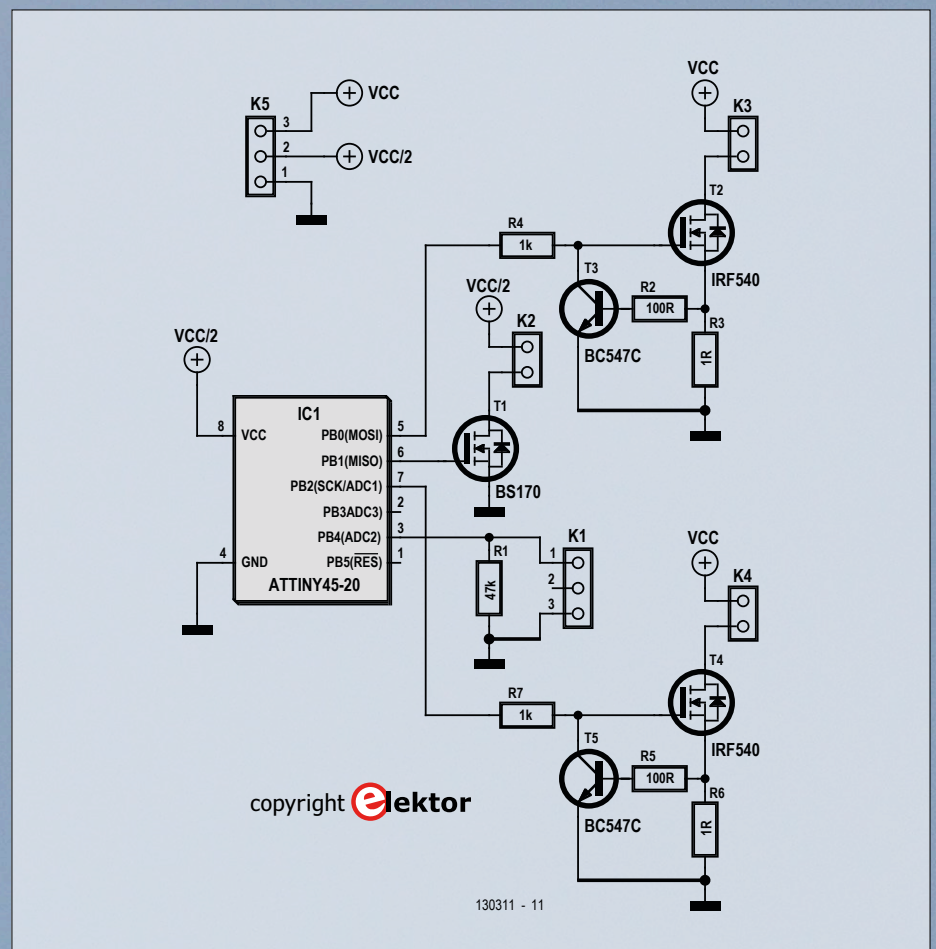
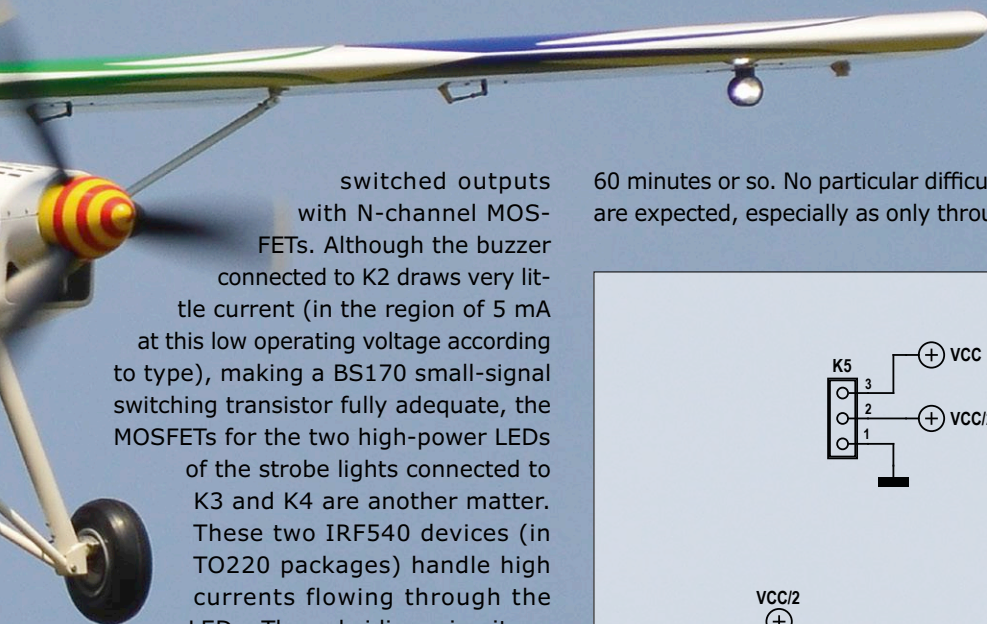


Figure 1. Schematic for model aircraft strobe lights.

Software for three modes

The software lets you operate the strobe light circuitry in three modes, which can be altered either during flight or after landing by means of a 3-position switch on the remote control.

- Switch backwards
Flashing lights and buzzer off.
- Switch central
Flashing lights on and buzzer off.
- Switch forwards
Flashing lights on and buzzer on.

The center setting is intended for use during flight, the third for afterwards. If your model should crash (we hope not but it happens even to top professionals), you can use the buzzer as an audible search tool to locate and

retrieve it easily, even in rough terrain or poor visibility. The note of the buzzer is set in the software around 4 kHz but you can change this easily to whatever suits you best.

The software [3] was developed first in BASCOM and ported later to C in the Arduino IDE. The decision as to which mode is active is made right at the start in the Main Loop and is dependent on the length of the servo pulse (smaller 1.25 ms, 1.25 to 1.75 ms and larger 1.75 ms). In mode 2, if LED flashes are required to provide information on the charge state of the battery, the voltage of the supply to the ATtiny45 is measured and evaluated. How simply this works can be determined from the program excerpt in **Listing 1**.

Battery state at a glance

Last autumn I installed two examples of the LED flasher in my two electrically powered gliders. This enabled me to establish that the flashing LEDs could be seen clearly in bright sunlight, good visibility and an estimated flying height of 100 meters (300 feet), even when the model was nearly 500 meters (1500 feet) distant. This was definitely not extrasensory perception, as it was confirmed by other model airplane flyers.

In flight tests at an elevation of around 30 m (100 feet), with the buzzer turned on, the sound was audible at a distance of up to 50 to 60 m (150 to 180 feet). Of course this does depend on whereabouts on the model you fit the buzzer and which type is used. At this low operating voltage of just over 3 V there are

Listing 1. V_{CC} measurement with the ATtiny45 (excerpt from program).

```
void measureBattery() {
    batteryVoltage = readVcc();
    if (batteryVoltage >= 3800) { //more than 3.7V: battery full
        ledBlink(3);
    }
    if (batteryVoltage >= 3500 && batteryVoltage < 3800) { //between 3.3V and 3.7V, apparently there is a
small offset
        ledBlink(2);
    }
    if (batteryVoltage < 3500) { //less than 3.4V: battery empty
        ledBlink(1);
    }
}

long readVcc() {
    // Read 1.1V reference against AVcc
    // first set the reference to Vcc and the measurement to the internal 1.1V reference
    ...
#ifdef __AVR_ATtiny25__ || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
    ADMUX = _BV(MUX3) | _BV(MUX2);
    ...
    delay(2); // Wait for Vref to settle

    ADCSRA |= _BV(ADSC); // Start conversion
    while (bit_is_set(ADCSRA, ADSC)); // measuring

    uint8_t low  = ADCL; // must read ADCL first - it then locks ADCH
    uint8_t high = ADCH; // unlocks both

    long result = (high << 8) | low;

    result = 1125300L / result; // Calculate Vcc (in mV); 1125300 = 1.1*1023*1000
    return result; // Vcc in millivolts
}
```




COMPONENT LIST

Resistors

R1 = 47kΩ
 R2,R5 = 100Ω
 R3,R6 = 1Ω
 R4,R7 = 1kΩ

Semiconductors

T1 = BS170
 T2,T4 = IRF540
 T3,T5 = BC547C
 IC1 = ATtiny45-20PU (DIP 8), programmed,
 Elektor Store # 130111-41

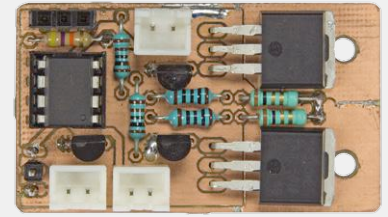
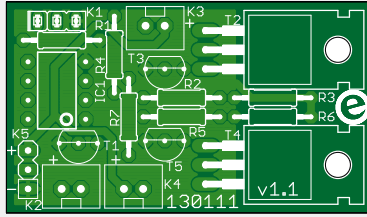


Figure 2. Component stuffing made easy: no SMDs in sight!

Miscellaneous

K1 = 3-pin pinheader, 0.1" pitch
 K2,K3,K4 = shrouded XH pinheader (JST from
 RS Components 820-1554) plus matching

socket (RS Components 820-1611) with
 crimp contacts (RS Components 820-1529)
 K5 = 3-pin pinheader, 0.1" pitch
 PCB 130111-1

Web Links

- [1] www.faa.gov/documentLibrary/media/Advisory_Circular/AC_20-30B.pdf;
www.buzer.de/s1.htm?g=luftvoanl_1&a=1-6; https://en.wikipedia.org/wiki/Navigation_light
- [2] http://cache.freescale.com/files/32bit/doc/app_note/AN4428.pdf
- [3] www.elektormagazine.com/130111

buzzers that whisper with just 72 dB and also those that produce a loud alarm tone of more than 105 dB under identical conditions! In this respect the acoustic range can be optimized by skillful selection.

Ideas for expansion

A little while back I built a larger lighting system for an aeromodelling friend. This enabled us to control additional LEDs, such as always-on red and green navigation lights as well as the intermittent ACL lights. Two white LEDs can be activated as strobe lights and a buzzer is also provided.

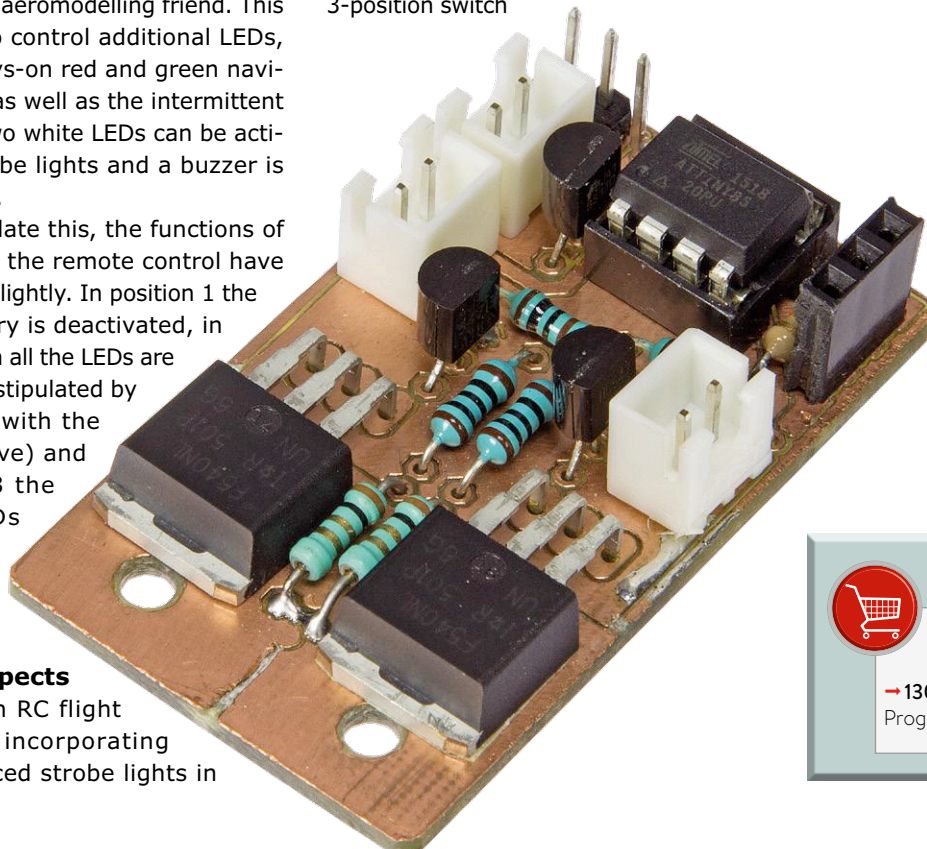
To accommodate this, the functions of the switch on the remote control have been altered slightly. In position 1 the strobe circuitry is deactivated, in center position all the LEDs are functional as stipulated by regulations (with the buzzer inactive) and in position 3 the flashing LEDs are off and the buzzer switched on.

Bright prospects

Right now an RC flight colleague is incorporating these enhanced strobe lights in

his electric glider with a wingspan of 4.5 meters (some 14 feet). The model's first test flight with the strobe indicator will follow soon afterwards. At the moment I am keeping busy by expanding the flasher to six functions. For this the flasher uses two servo connections on the receiver. At the transmitter we use one 3-position switch

plus a slider switch for controlling the flasher and the other functions. I would be very pleased to receive feedback from anyone who replicates these flashers in their various versions. Further suggestions and improvement ideas would be extremely welcome too. ◀
 (130111)



FROM THE STORE

→ 130111-1
 Bare PCB

→ 130111-41
 Programmed microcontroller



ELEKTOR ETHICS

Smart Devices

What do you get with them?

By Tessel Renzenbrink

The door refused to open. It said, "Five cents, please." He searched his pockets. No more coins; nothing. "I'll pay you tomorrow," he told the door. Again he tried the knob. Again it remained locked tight. "What I pay you," he informed it, "is in the nature of a gratuity; I don't have to pay you."

"I think otherwise," the door said. "Look in the purchase contract you signed when you bought this conapt."

In his desk drawer he found the contract; since signing it he had found it necessary to refer to the document many times. Sure enough; payment to his door for opening and shutting constituted a mandatory fee. Not a tip.

"You discover I'm right," the door said. It sounded smug.

From the drawer beside the sink Joe Chip got a stainless steel knife; with it he began systematically to unscrew the bolt assembly of his apt's money-gulping door.

"I'll sue you," the door said as the first screw fell out.

– Philip K. Dick, *Ubik*.

What do you actually get when you buy a smart device? Are you giving uninvited guests access to your home when you replace your dumb devices by smart devices? In the science fiction novel *Ubik* by Philip K. Dick, the main character Joe Chip is constantly running into smart objects that demand something from him before he can use them. That was a fanciful notion when the book was published in 1969, but now in 2017 it looks prophetic.

The vengeful maker

The modern equivalent of Dick's fantasy is the spat which broke out between the maker of Garadget and a user of the device in April this year. That IoT device allows you to open a normal garage door via the cloud. The user, who identified himself or herself as R. Martin, was not able to get the device working and asked the Garadget community for help. When user Martin did not receive immediate assistance, he or she posted a negative review on Amazon.com with the words "a piece of junk." The maker of Garadget did not take that kindly and retaliated by disabling the device. That was possible because the device is operated via a cloud platform which is controlled by the device

maker. On the forum the maker sent the following message to user Martin: "Your device with ID number 2f0036... has been denied access to the server. Your only option is to send the device back to Amazon.com and request a refund."

A web of relationships

With the advent of smart devices, a fundamental change has taken place in the relationship between producers and consumers. In the traditional scenario, buying a product is a one-time transaction: you buy a refrigerator, have it delivered to your home, and plug in the power cord. During the next ten years you don't have to give a second thought to the refrigerator, and you have no further contact with the vendor or the manufacturer.

By contrast, with smart products you enter into a long-term relationship with the manufacturer and/or vendor. Instead of buying a product, you purchase a product service package. Part of this service, for example, could be maintaining the cloud platform that supports your smart refrigerator. This means you have to trust that the refrigerator manufacturer will actually keep the server up and running for those ten years. In many cases there is more than one party involved, resulting in a complex web of relationships. For their part, the manufacturer is dependent on the external developer of the cloud platform. They have to stay solvent for the duration of the relationship. You are also dependent on regular firmware updates to ensure that your refrigerator does not become unsafe or unusable. This already happens now with many smartphones, which despite having perfectly serviceable hardware become obsolete because the operating system is no longer supported.

And what happens with the data that your device collects? Recently the smart TV manufacturer Vizio had to pay a hefty fine because they sold the viewing profiles of 11 million customers to another party. Incidentally, the fine was not imposed simply because they sold the data, but because they did not first inform the viewers.

Tractors standing still amidst wilting crops

There are many examples of consumers who are at the mercy of smart devices or the producers of smart devices. The main loss of the Garadget user is limited to the Saturday afternoon they spent installing the device, but in other situations the consequences can be much more significant.

The time when the tractors made by the US farm machinery manufacturer John Deere were simple vehicles with internal combustion engines is long past; now they are mobile computers with hardware and software components. When farmers buy these tractors, they have to sign a conditions of use agreement. It stipulates that the owner is not allowed to make

any repairs to their vehicle, but instead must have this done by a certified repair firm. Along with being more expensive, the official repair firms are often further away, so it takes longer before the vehicle is again operational. During the harvest season, every minute that the machine is out of service costs a lot of money.

The time when American farmers maintained their own machinery, with the assistance of the local car mechanic where necessary, is long gone. At least if they want to do it legally. For the tech site Motherboard, journalist Jason Koebler spoke with farmers who are seeking help from hacker forums where they can buy hacked software for maintaining their machines [1]. For example, on those sites they can download a diagnostic program that identifies hardware and software problems. In the opinion of the farmers, John Deere should simply make this program available to them when they buy the tractor. With the hacked software the farmers regain a bit of control and independence with respect to their implements, but that comes at a price. The farmers are afraid that John Deere will come after them for violation of the conditions of use. They told Koebler

that their greatest fear is that the manufacturer could remotely disable their tractor, leaving it useless for them. Just like the smug door in the story about Joe Chips, this shows that smart devices may be intelligent but it is important to ask who benefits from this intelligence. ◀

(160393-1)

Web Links

[1] <https://is.gd/mpeZUY>



A puppeteer (photo: public domain).



welcome in your **ONLINE STORE**

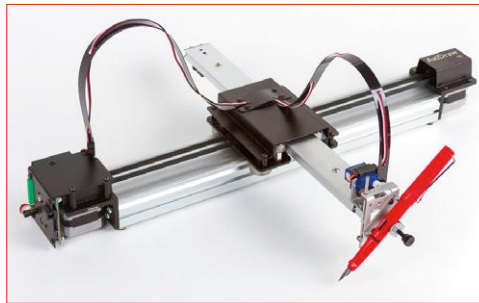
EDITOR'S CHOICE



AxiDraw is a drawing table or plotter, or in classical terms, a drawing robot, capable of drawing and writing on any flat surface. Not just paper, but also cardboard, whiteboards, large panels, sheet metal, fabric on a frame, etc. By comparison with a conventional table plotter, AxiDraw's great originality is being able to draw on surfaces that are bigger than the machine itself. AxiDraw is not a kit, it is supplied (almost) ready to use. It is not a toy, more a tool for experimentation and (re)production. Its construction of right-angled XY sliders is robust and rigid (extruded aluminum); all the rigid parts are metal, even the pen carrier and the end caps. The impression of great precision is confirmed by the weight

(2.2 kg or nearly 5 lb.). The AxiDraw is an extremely versatile machine, designed to serve a wide variety of everyday and specialized drawing and writing needs!

Denis Meyer (Elektor Labs)



www.elektor.com/axidraw-v3

Elektor Bestsellers

1. SmartScope
www.elektor.com/smartscope



2. Digital Microscope V160
www.elektor.com/microscope-v160
3. GSM/GPRS Projects
www.elektor.com/gsm-gprs-projects
4. Raspberry Pi 3 (Model B)
www.elektor.com/rpi3
5. Mastering Microcontrollers Helped by Arduino
www.elektor.com/mm3
6. DVD Elektor 2016
www.elektor.com/dvd-elektor-2016
7. BBC micro:bit (Book)
www.elektor.com/microbit-book

Mastering Microcontrollers Helped by Arduino



This book will not only familiarize you with the world of Arduino but it will also teach you how to program microcontrollers in general. Having completed this fun and playful course, you will be able to program any microcontroller, tackling and mastering I/O, memory, interrupts, communication (serial, I²C, SPI, 1-wire, SMBus), A/D converter, and more. This third, extended and revised edition contains two new chapters: AVR Playground and Elektor Uno R4.

member price: £32.95 • €38.66 • US \$43

www.elektor.com/mm3

Retro Audio

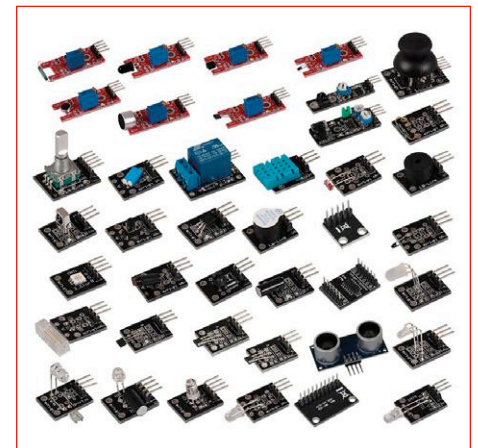


Today there is a re-emerging, nostalgic interest in vinyl records and associated music entertainment gear. With this interest, there is a paralleled market for the repair of this gear. The intention of this book is to offer the reader understandings, ideas and solutions from the perspective of a workbench technician and electronics hobbyist. It is a descriptive text with many tables of useful data, servicing tips and supplementary notes of not so common knowledge.

member price: £22.95 • €26.95 • US \$30

www.elektor.com/retro-audio

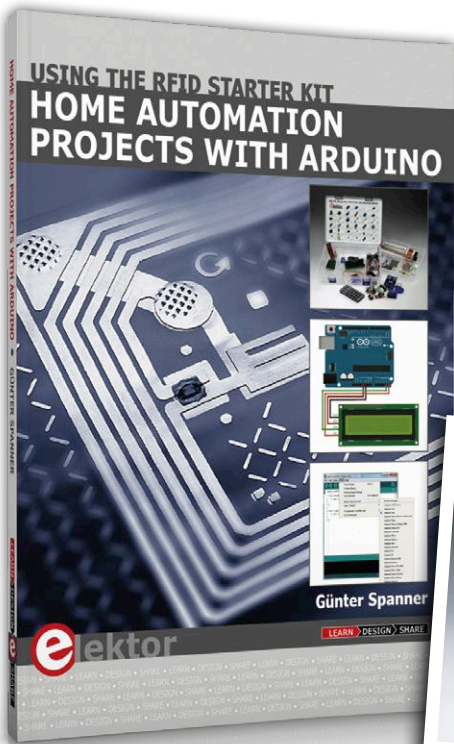
Sensor Kit X40



This high-quality sensor kit has been specially developed for the most common open-source platforms. It is compatible with single-board computers (Raspberry Pi, Banana Pi, CubieBoard, CubieTruck, Beaglebone, pcDuino) and microcontrollers (Arduino, ATmega, AVR, PIC, STM32, etc.). It contains a total of 40 different sensors. You can either solder the sensors or stick them on a board to work on different circuits or experiments.

member price: £60.95 • €71.96 • US \$79

www.elektor.com/sensor-kit-x40



Home Automation Projects with Arduino

Package Deal: Book + RFID Starter Kit



If you really want to build some innovative projects it's often necessary to get down to component level. This can present many beginners with major problems. That is exactly where this book begins. It explains how a wide variety of practical projects can be built using items supplied in a single kit together with the Arduino board. This kit, called the 'RFID Starter Kit for Arduino Uno' is not just limited to RFID applications but contains more than 30 components, devices and modules covering all areas of modern electronics.



MEMBER PRICE: £59.58 • €69.66 • US \$77.31
www.elektor.com/rfid-bundle

StromPi 2



The StromPi 2 expansion board gives your Raspberry Pi the flexibility to be deployed in a much wider range of environments such as vehicular, marine, truck and industrial plant etc. On top of this, the Raspberry Pi USB outputs can be optionally upgraded to a high-power USB port; this allows the system to use power-intensive peripherals such as USB hard drives without limitation.

member price: £22.95 • €26.96 • US \$30

www.elektor.com/strompi-2

GSM/GPRS Projects



Every mobile phone in the world has a GSM/GPRS modem which enables the phone to communicate with the world outside. These modems enable the phone to establish audio conversations, send and receive messages and connect to the Internet. This book is aimed at people who want to learn how to use the GSM/GPRS modems in microcontroller based projects. Two types of popular microcontroller families are considered in the book: PIC microcontrollers, and the Arduino.

member price: £34.95 • €26.95 • US \$30

www.elektor.com/gsm-gprs-projects

Sand Clock



This supercool gadget built around an Arduino Uno writes the time into a layer of sand. After an adjustable time the sand is flattened out by two vibration motors and everything begins all over again. This kit contains everything to build this sand clock by yourself: all mechanical parts, the motors, an Arduino Uno, a special RTC/driver shield, a power adapter and even a small bag with sand.

member price: £91.95 • €107.96 • US \$119

www.elektor.com/sandclock

Hexazener

a mind-teaser for elektornics enthusiasts

On the occasion of glorious summertime the Hexadoku grid got replaced with a "Hexazener" grid (notice the grid's outline resembling the zener diode symbol). The rules for filling in the grid and the characters allowed are the same as for the normal Hexadoku grid.

Reminder: a Hexadoku grid uses figures from the hexadecimal system, i.e. from 0 to F. You should fill it in so that all the hexadecimal figures from 0 through F (0-9 and A-F inclusive) are used once and only once in each row, column, and block (identified by a thicker line).

The Hexazener differs from the Hexadoku in the following ways:

- The grid has 16 blocks, but 18 lines and 20 columns.
- Four areas are empty (white).
- The blocks are not square or rectangular, but are more like

pieces of a jigsaw (referred to as a hexadoku jigsaw).

- The number of boxes in each block depends on its shape and position, but is never more than 16. So this grid can still be described as a Hexadoku.
- The total number of active boxes is 186, compared to 256 for a Hexadoku.

As certain blocks contain fewer than 16 boxes, it's not possible to use all the hexadecimal characters in these ones. This is just a little extra difficulty to bear in mind when solving it.

Certain figures have already been entered in the grid to define the starting situation. In order to enter, there's no need to send us the whole grid, all you need do is send us the sequence of six figures in the grey boxes.

Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for five Elektor Book Vouchers worth **\$70.00 / £40.00 / €50.00 each**, which should encourage all Elektor readers to participate.

Participate!

Ultimately July 21, 2017, supply your name, street address and the solution (the numbers in the gray boxes) by email to:

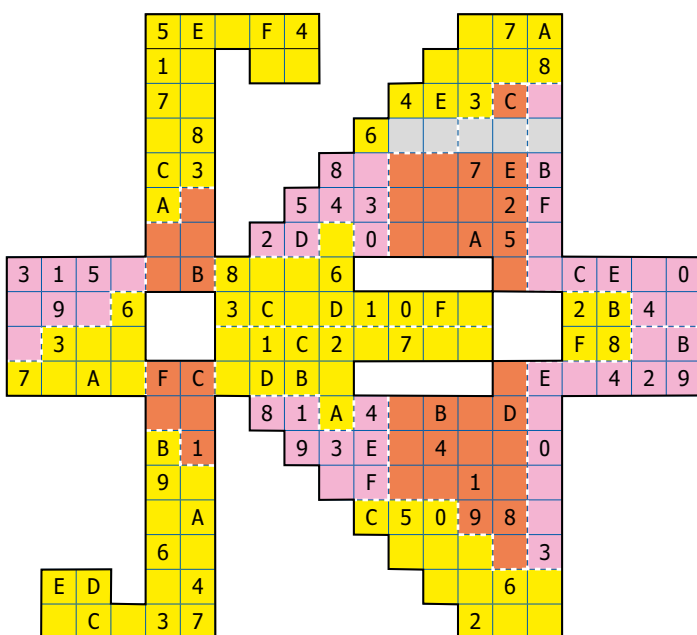
hexadoku@elektor.com

Prize Winners

The solution of Hexadoku in edition 3/2017 (May & June) is: **1E709**.

The €50 / £40 / \$70 book vouchers have been awarded to: Vincent Pierdominici (USA); Ariaan Hoed (Belgium); Nikos Chararas (Austria); Guy Savage (UK); and Jean Thevenoux (France).

Congratulations everyone!



2	4	8	C	F	5	B	3	6	A	1	E	7	0	9	D
7	B	D	E	0	2	4	6	8	9	3	F	A	C	5	1
5	9	6	3	7	A	D	1	B	C	0	2	4	8	E	F
F	0	1	A	C	8	9	E	7	D	4	5	2	3	6	B
1	7	5	9	8	D	C	F	3	E	6	0	B	4	A	2
0	C	2	4	6	B	A	9	5	F	D	8	3	7	1	E
8	F	A	D	E	3	7	4	C	B	2	1	5	6	0	9
B	E	3	6	1	0	2	5	4	7	9	A	8	D	F	C
3	2	7	F	B	E	5	8	A	0	C	9	6	1	D	4
4	5	B	8	9	1	0	7	D	2	E	6	C	F	3	A
9	A	C	0	D	6	F	2	1	3	B	4	E	5	7	8
D	6	E	1	4	C	3	A	F	5	8	7	9	B	2	0
E	8	9	5	3	4	1	C	0	6	A	D	F	2	B	7
C	1	F	B	A	7	E	0	2	4	5	3	D	9	8	6
6	3	4	7	2	9	8	D	E	1	F	B	0	A	C	5
A	D	0	2	5	F	6	B	9	8	7	C	1	E	4	3

The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.

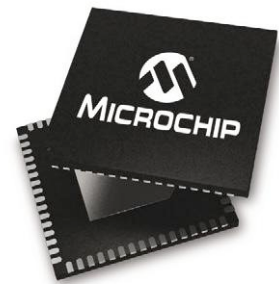
PIC18F "K40" MCUs feature Intelligent ADC with Filtering and Signal Analysis Capabilities

8-bit PIC® MCUs are ideal for Touch and Signal Conditioning



The Core Independent Peripherals (CIPs) in Microchip's PIC18F "K40" family of 8-bit PIC® MCUs support filtering and signal analysis for advanced touch and signal-conditioning applications.

Among the intelligent analog CIPs is an Analog-to-Digital Converter with Computation (ADC2) for averaging, filtering, oversampling and automatic threshold comparison. The MCUs also integrate safety-critical CIPs and hardware PWMs with multiple communication interfaces and generous on-chip Flash and EEPROM. These features, combined with 5V operation, enable the PIC18F "K40" family to increase design flexibility whilst also reducing system cost.



microchip
DIRECT
www.microchipdirect.com

 **MICROCHIP**

www.microchip.com/EUPIC18FK40

The Microchip name and logo, the Microchip logo, MPLAB and PIC are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. All other trademarks mentioned herein are the property of their respective companies. © 2016 Microchip Technology Inc. All rights reserved. DS30010138A. MEC2125Eng11/16

DO YOU WANT THE BEST ELECTRONICS DESIGN SOFTWARE



FEATURES

- Schematic Capture
- PCB Layout
- Gridless Autorouting
- 3D Visualization
- M-CAD Integration
- SPICE Simulation
- MCU Co-simulation
- Built in IDE
- Visual Programming

NOW INCLUDES:

Serpentine Routing, Layer Stackup Manager, and Assembly Variants.

labcenter  www.labcenter.com

Electronics

(+44) 01756 753440